|  |
| --- |
| Advance Data Science |
| Boston Energy Forecasting |
| Mid Term |

|  |
| --- |
| Lalit Jain, Sameer Goel, Lipsa Panda – Team 7  Date: 11-18-2016 |

Contents

[1 Executive Summary 2](#_Toc467267133)

[2 Data Ingestion and Wrangling 2](#_Toc467267134)

[2.1 The Data 2](#_Toc467267135)

[2.2 Feature Enhancement 3](#_Toc467267136)

[2.3 Getting Weather information 5](#_Toc467267137)

[2.4 Merging Data with Weather Data 8](#_Toc467267138)

[3 Part 2: Modelling 12](#_Toc467267139)

[3.1 Required fields 12](#_Toc467267140)

[3.2 Generic approach: Prediction 12](#_Toc467267141)

[3.2.1 Feature Selection 12](#_Toc467267142)

[3.2.2 Creating the Function and reading the input 13](#_Toc467267143)

[3.2.3 Conversion of data type 13](#_Toc467267144)

[3.2.4 Split Validation and Scaling the Features 13](#_Toc467267145)

[3.2.5 Baseline Model 13](#_Toc467267146)

[3.2.6 Different Machine Learning Algorithms and Output 14](#_Toc467267147)

[3.2.7 Storing and returning the results 16](#_Toc467267148)

[3.2.8 Meta Evaluator 16](#_Toc467267149)

[3.2.9 Prediction Model Evaluation: Which model to choose 17](#_Toc467267150)

[3.2.10 OutLier Analysis Random Forest 17](#_Toc467267151)

[3.3 Generic approach: Classifcation 18](#_Toc467267152)

[3.3.1 Different Machine Learning Algorithms and Output 18](#_Toc467267153)

[3.3.1.3 Random Forest 19](#_Toc467267154)

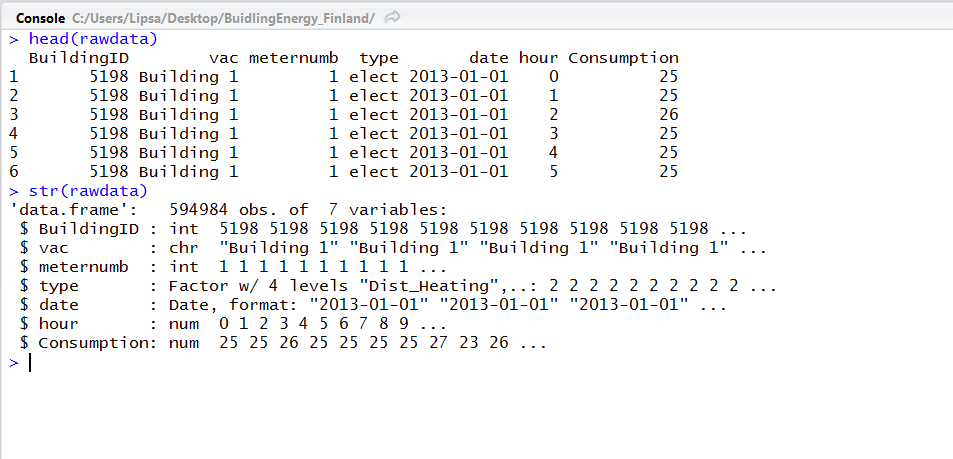
# Summary

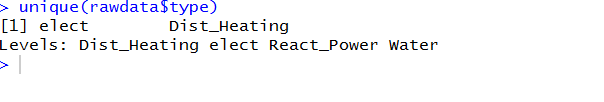
The report summarizes the energy usage by the buildings owned by Vokia Inc. The purpose of the report is to document the Energy models and the subsequent analysis to understand and reduce energy usage to make the building energy efficient.

# Part1: Data Ingestion and Wrangling

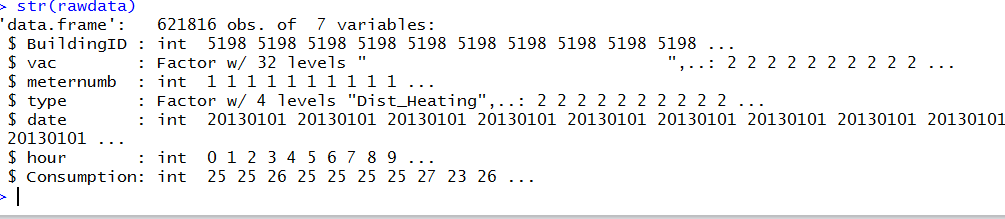
## The Data

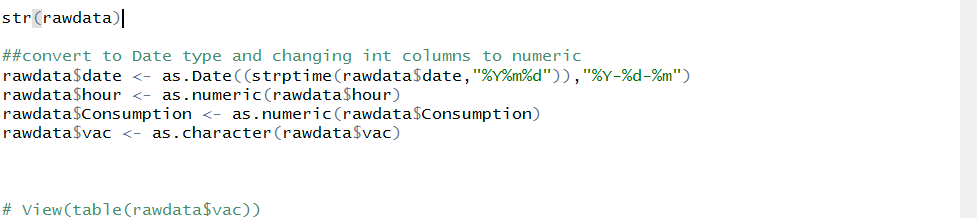
* R file available in Part 1/Part1\_Ingestion.R
* We have a data set with 1214184 records with corresponding building details. We can have an idea about the data and the data types of the variable used.

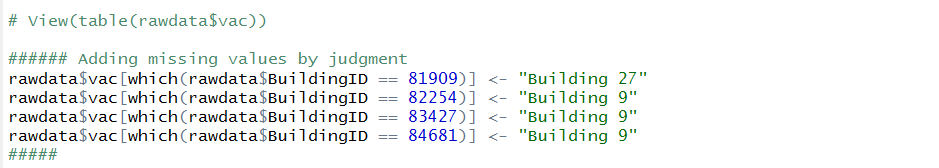


* We found the raw data had 4 different types of sources of energy out of which only 2 were our matter of concern. 
* Hence, we filtered the data set to fetch the energy consumption by Electricity and Heating. Once filtered, we read the data from dataset, R intuitively understands datatypes. After doing a filter with the required source types, the number records came down to 621816.

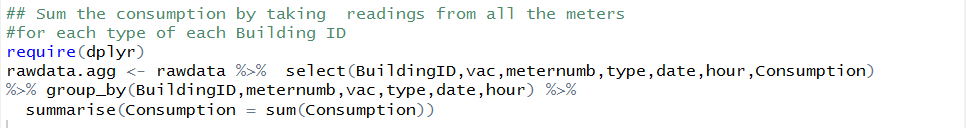
We need to check the structure of our data and then change the columns structure according to our requirements. str () function gives us the snapshot of datatypes of all the columns.



* There were few values like Date, Hour, vac which needed to be converted to corresponding formats.
* After having an insight on the data, we analyzed there were few missing building records which we substituted with the corresponding building numbers.

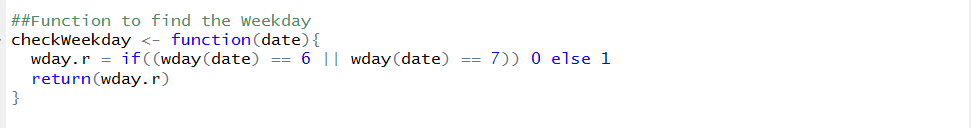


* We added the energy consumption by taking the reading from all the meters for each building to have a confirmation that the each building with corresponding building id has a meter id reading on an hourly basis.

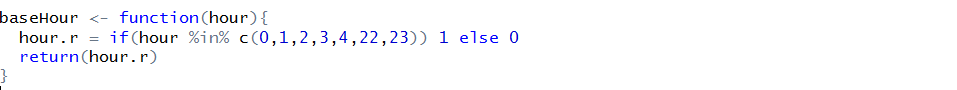


## Feature Enhancement

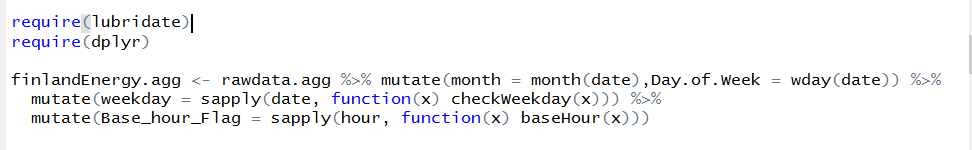
* Now adding features like week of the day, Month of the year, Weekday or Weekend, Holiday and Base Hour Flag.
* Function to find the weekday for the data record:



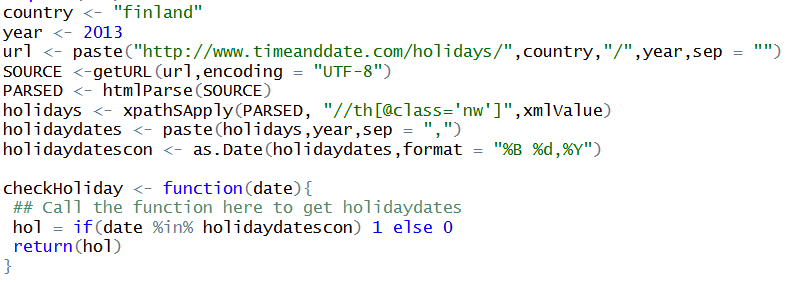
* We used another function to calculate the base hour and set the values high for the duration 0,1,2,3,4,22,23.



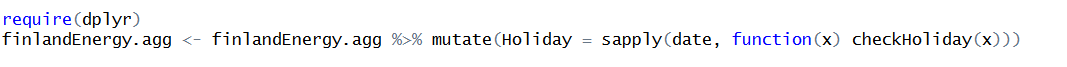
* To work with date-times and time-spans: fast and user friendly parsing of date-time data, extraction and updating of components of a date-time (years, months, days, hours, minutes, and seconds), algebraic manipulation on date-time and time-span objects. The **'lubridate'** package has a consistent and memorable syntax that makes working with dates easy and fun. So, we used the following package.
* Using the above package, we derived a dataset with the above values using mutate function.

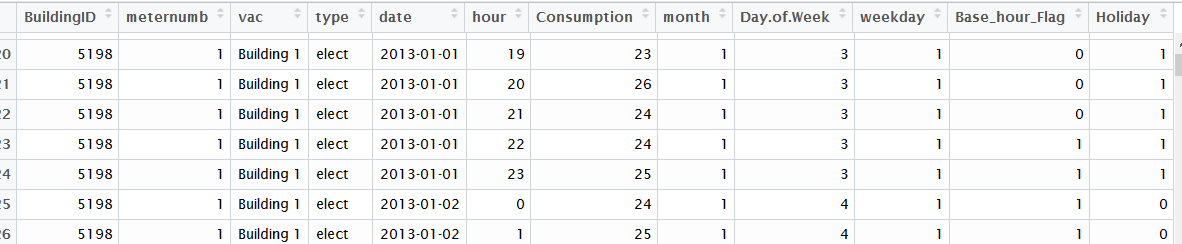


* After extracting the required features, we normalized the consumption with respect to the area\_floor\_m.sqr and made it Kwh/square meter.
* To get public holiday details, we scraped the timeanddate.com website. Code is as shown below. All the holidays are stored in holidaydates objects. We also defined a function which can take any date passed to it and it will return if the date is holiday or not.



* We will now call the pass our data set and append a Holiday column to the dataset. Column will be evaluated using the above function defined



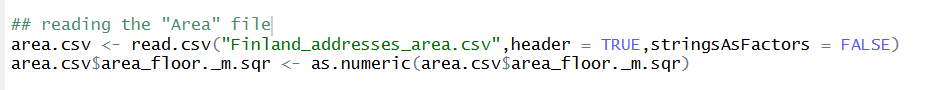


## Getting Weather information

* We will read the other file, which contains the address information of each building. To get the Weather data, we will follow the below approach
  + Using the address, find the latitude and longitude of the building using Google API. It will return the latitude and longitude of the address
  + Using the latitude and longitude information we will find the nearest city (airport code) associated to it by using wunderground api.
  + Using the nearest city, we can use wunderground library to get the Weather information related to the particular building.
* We have defined few functions to solve the above purpose:

1. To get the Latitude & Longitude by sending the address (GeoCode.R)
2. Getting the nearest city to the airport(AirportFunction.R)
3. Fetching the weather data for each airport code (WeatherFunction.R)

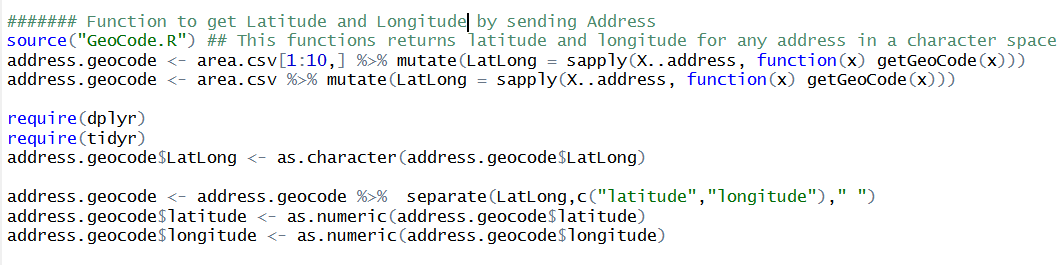
* First we will read the address information using the csv file

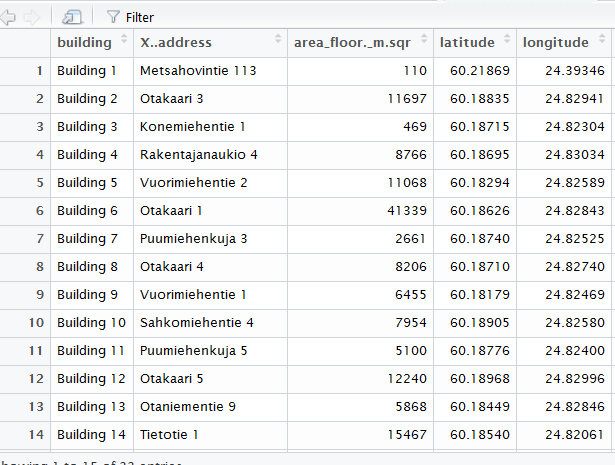


* To get the Latitude & Longitude, we defined the function as shown below. **getGeoCode function** will calculate the longitude and latitude based on the address provided to the function from the google API. The output has Longitude as lat and Longitude as lng.

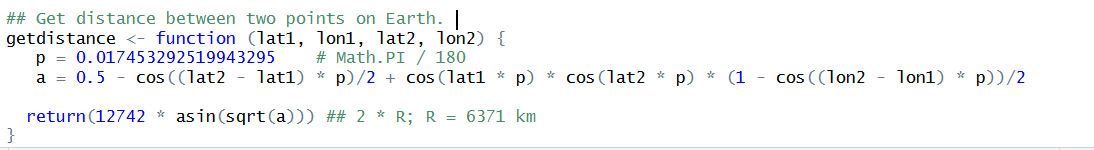


* The function returns the latitude and longitude as character without space which we have then made two columns of numeric type.
* Since, we have the function ready, we can call this function on our area.csv and append two columns of Latitude and Longitude next to it using the below code

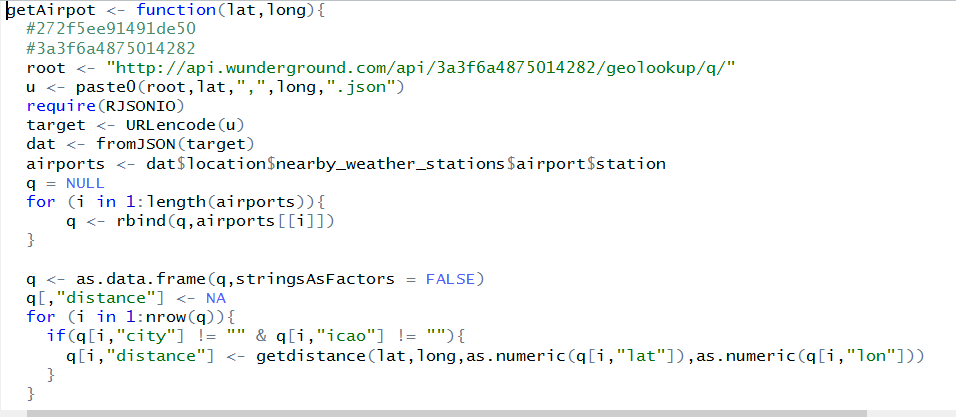




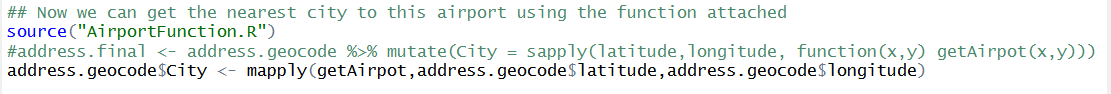
* The second function **Airport Function** would fetch the closest city using the wunderground api. The call to the api gives us sometimes more than one nearest city associated with the latitude and longitude.
* We can handle this by calculating the distance between the actual latitude and longitude and the output latitude longitude using the function getDistance as defined below



* Actual function looks like below. It takes two argument as input lat and long and call the api based on the received lat and long. Received JSON data is parsed and the result is saved in a dataframe. For each record in the dataframe, getdistance method is called to calculate the distance. At last, dataframe containing the least distance is sent back with the city code.

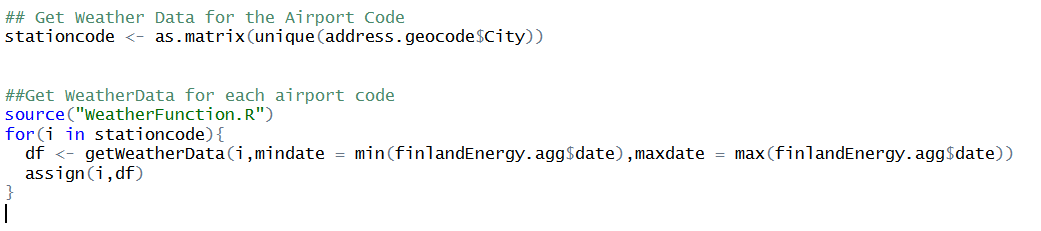


* Since we have the function defined now, we can call the function using the below code which will give us our required airport code



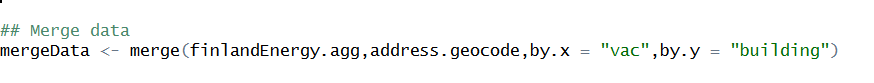


* Now we can define our last Weather code function which receives a city code, min and maximum date and returns a data frame with the weather information using the wunderground library. **Please refer WeatherFunction.docx for the function**
* Since we do not want to call the Weather Function 33 times (number of records) we can find unique airport code and call the weather function and save the weather data for each station code

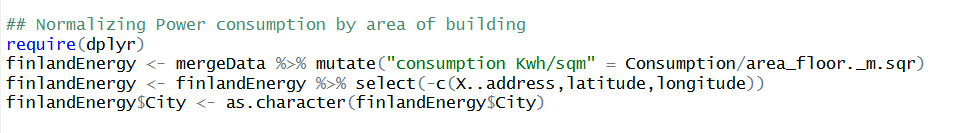


## Merging Data with Weather Data

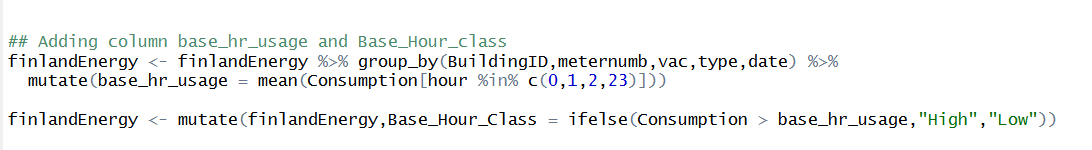
* Now since we have both the raw data and weather data, we can merge the dataset together respectively. Before that we need to normalize the rawdataset and make some changes
* Adding address information to the original dataset

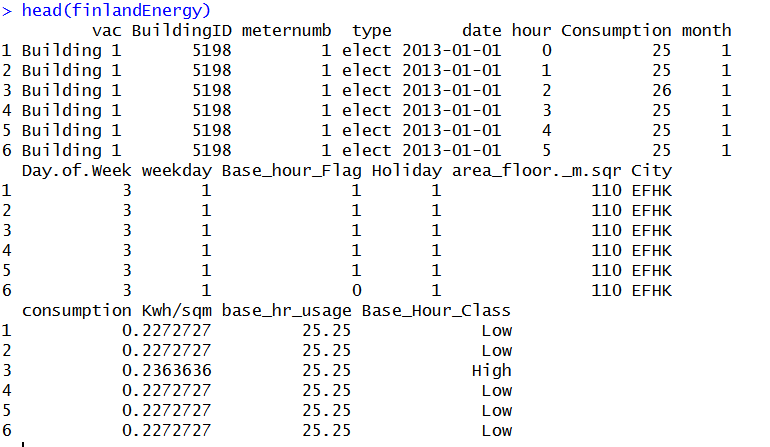


* We will now add a normalized consumption column which will be consumption divided by the area of the building. We can name it “Consumption Kwh/Sqm”.
* Followed up by removing unwanted columns

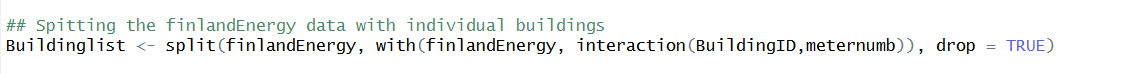


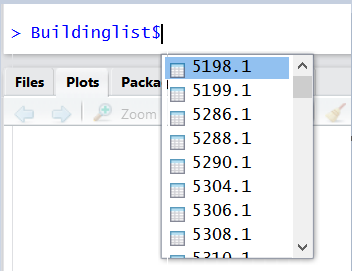
* Adding Base\_Hr\_Usage and Base\_Hour\_class column by average the sum of the consumption for hours in 0,1,2, and 23 and later a binary column which will be “High” if the consumption is greater than base\_hr\_usage and vice versa



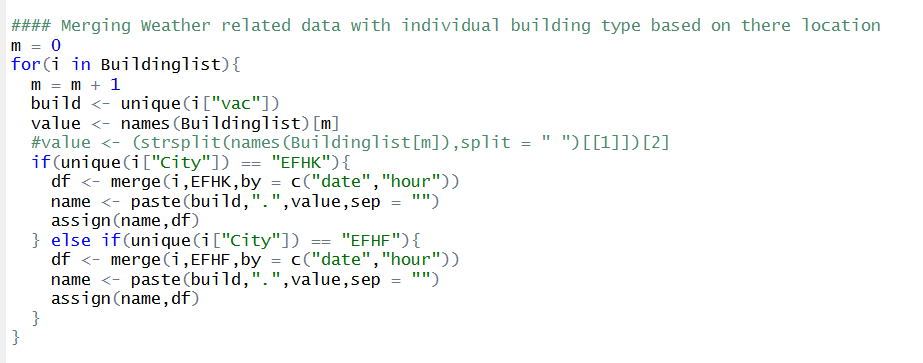
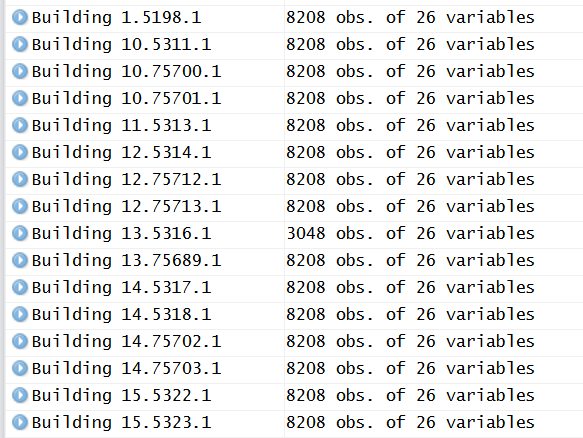


* Before adding the weather data to our ‘finlandEnergy’ dataset. We need to divide our dataset with individual buildings and then add respective weather data.
* We will use split function which can divide the data frame into a grouped dataframe, which contains the set of dataframes with the combination of Building ID and meternumber.

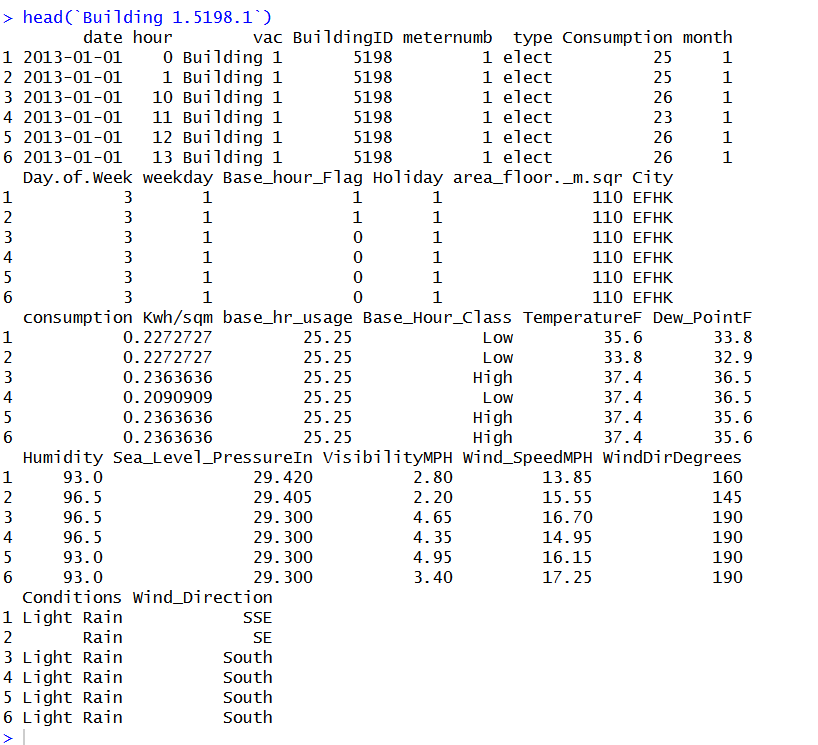




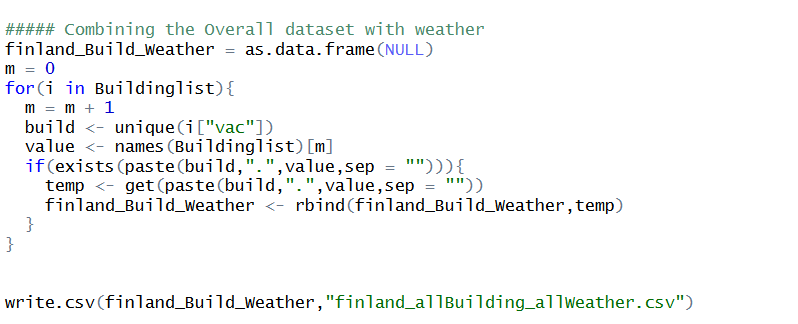
* Now we can loop through our BuildingList and merge the respective weather information as shown in the below code. We can see individual datasets being created in the gobal environment with their respective weather data



* Checking a dataframe to ensure the information is as desired



* We can now merge all the individual dataset to a single dataframe and use that for analysis.



# Part 2: Modelling

## Required fields

* We have added Base\_hour\_Usage and Base\_Hour\_Class columns in the original dataframe as mentioned in Part 1

## Generic approach: Prediction

We will predict consumption (KWH) for our Finland Energy Dataset using Regression, KNN, Random Forest and Neural Network models below. We will calculate predictions for each 78 Buildings we have and using all the models.

Approach for the Prediction is as follows

* We will create functions which can take a csv or object and return the coefficients of the model
* A for loop will be called which will call the defined function and give the results back
* Results will be appended and can be written down in a csv file

**Note: For the functions and the result, please refer the following files in the Git repository**

1. **Part2/SingleBuilding\_Prediction/**

### Feature Selection

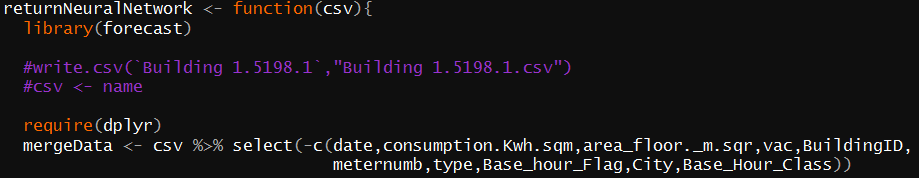
Before proceeding with our models we have done best feature selection using four algorithms exhaustive search, forward, backward selection, stepwise regression. The best features that add to the predictive power of the model and irrelevant features removed from the model.

Performing all the feature selection methods we shortlisted below features to best predict our model.



### Creating the Function and reading the input

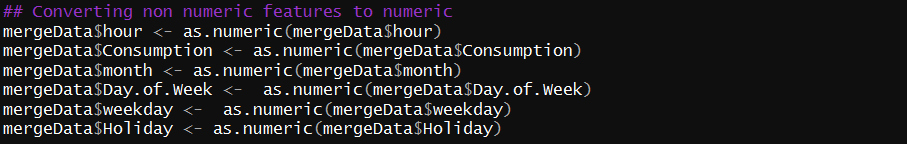
We have created functions that would take csv as a data frame or csv file.



Csv data inserted into mergeData after removing the unwanted features.

### Conversion of data type

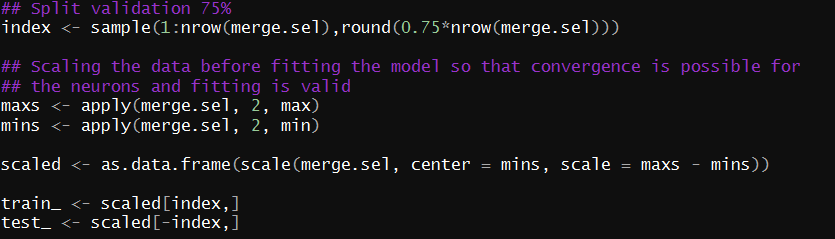
Data types of the read file is converted as below.



### Split Validation and Scaling the Features

We are using the split validation and scaling of features. When you suspect that the data is not consistent. You can easily see this when you go through the results of the summary() function.

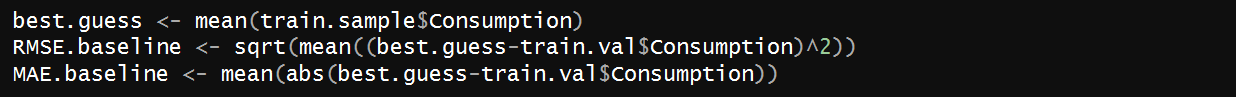
Look at the minimum and maximum values of all the (numerical) attributes. If you see that one attribute has a wide range of values, you will need to normalize your dataset, because this means that the distance will be dominated by this feature.



With above function we have computed maxs and mins of the numerical features and excluding the categorical features and then re-distributed the values.

### Baseline Model

In the absence of any predictor, all we have is the dependent variable (Consumption). What would be our best guess if we had to predict the amount of Consumption, on a given day, in the test set? It is the mean of the Consumption values, in the training data, is the best value.

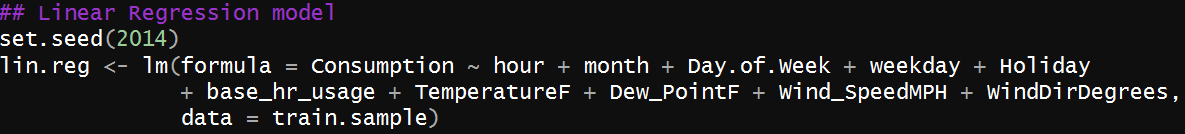


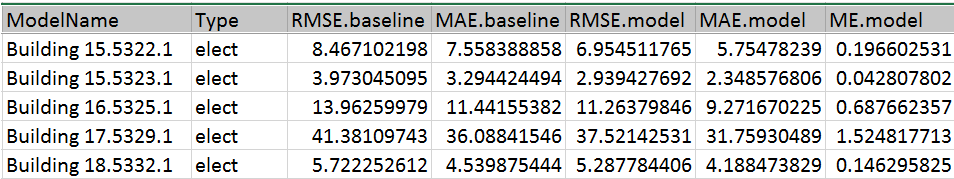
This value will give us the fair idea on how are the predictions compared to RMSE baseline and RME baseline, in further steps.

### Different Machine Learning Algorithms and Output

#### Regression

Linear regression is an approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables.

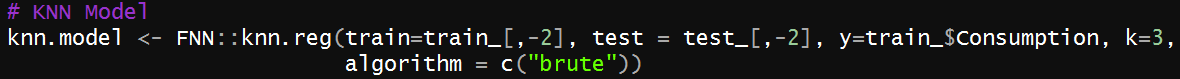


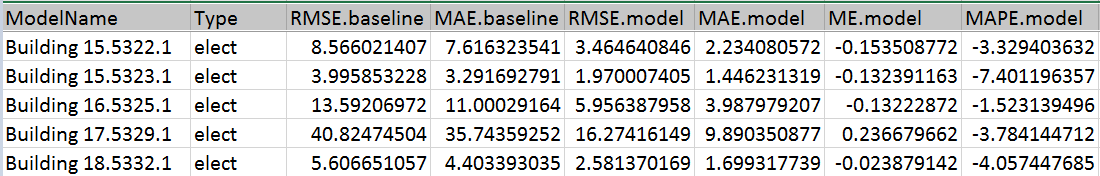


#### KNN

KNN or k-nearest neighbors’ algorithm is one of the simplest machine learning algorithms and is an example of instance-based learning, where new data are classified based on stored, labeled instances. More specifically, the distance between the stored data and the new instance is calculated by means of some kind of a similarity measure.

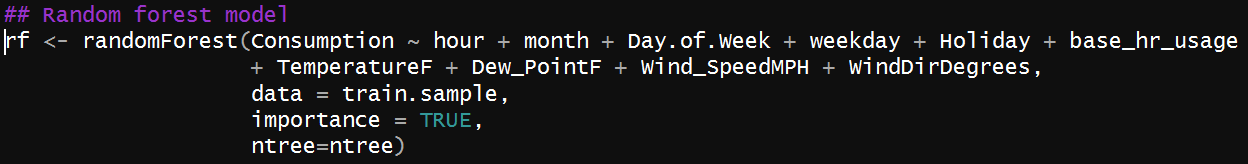
This similarity measure is typically expressed by a distance measure such as the Euclidean distance, cosine similarity or the Manhattan distance.

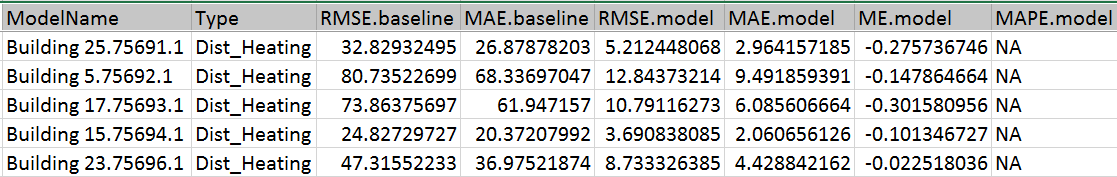




#### Random Forest

The random forest starts with a standard machine learning technique called a “decision tree”. This is a type of additive model that makes predictions by combining decisions from a sequence of base models.

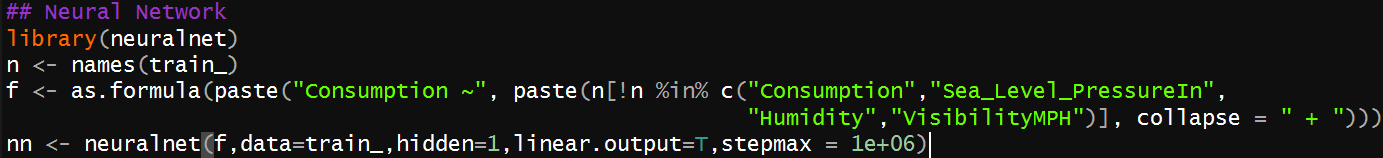


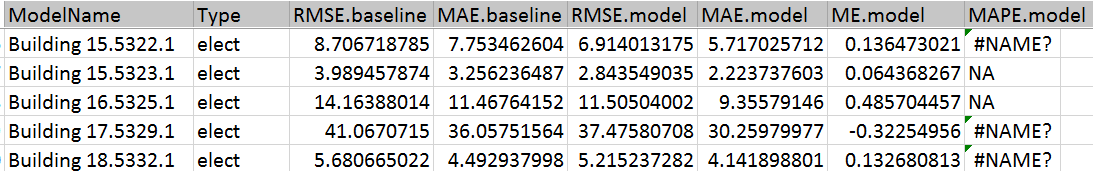


#### Neural Network

Neural network terminology is inspired by the biological operations of specialized cells called neurons. A neuron is a cell that has several inputs that can be activated by some outside process.

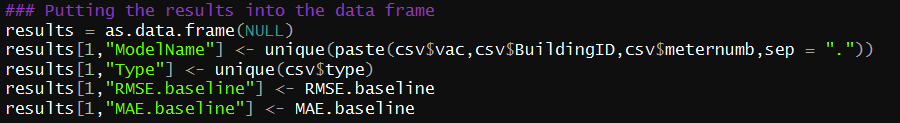
The artificial equivalent of a neuron is a node (also sometimes called neurons, but I will refer to them as nodes to avoid ambiguity) that receives a set of weighted inputs, processes their sum with its activation function, and passes the result of the activation function to nodes further down the graph.



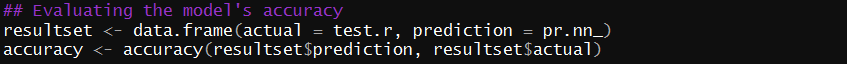


### Storing and returning the results

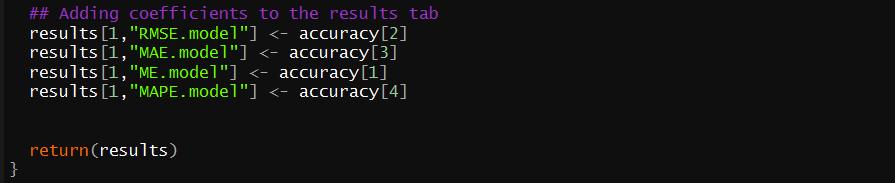
We are storing the results as below, where ModelName, Type, RMSE.baseline, MAE.baseline.



Calculating the results from the different machine learning algorithms.



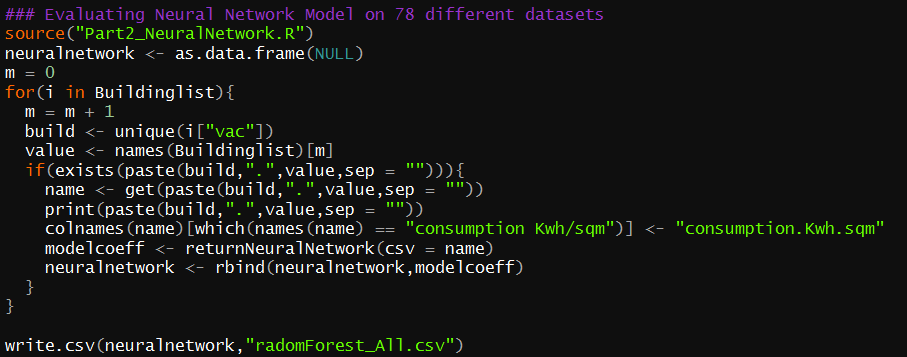
When we have all the output values, we will return it from the function



### Meta Evaluator

The above function is run repeatedly for all the datasets using below code. We will execute 4 ‘for’ loops for every Algorithm for 78 different datasets which computes the MAPE, MAE, RMSE values along with Baseline values.

Below is the sample code for Neural network ‘for’ loop, we have similar loop for Regression, KNN and Random Forest.

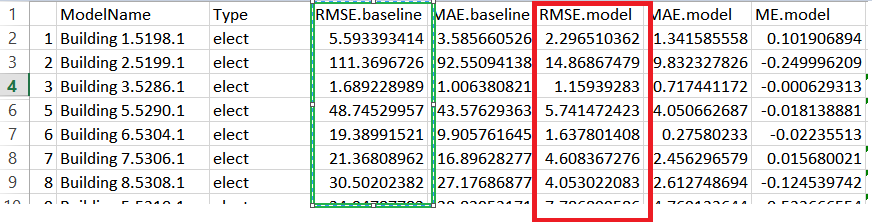


**Note: For the functions and the result, please refer the following files in the Git repository**

1. **Part2/SingleBuilding\_Prediction/** **Part2\_ModelEvaluation.R**

### Prediction Model Evaluation: Which model to choose

After we have the output from all the files for all the model, the baseline RMSE value will give fair idea on how are the predictions compared to RMSE value of the model. We had compared results of all the models decided **Random Forest** gives us much better result. Random forest output is as shown below



### OutLier Analysis Random Forest

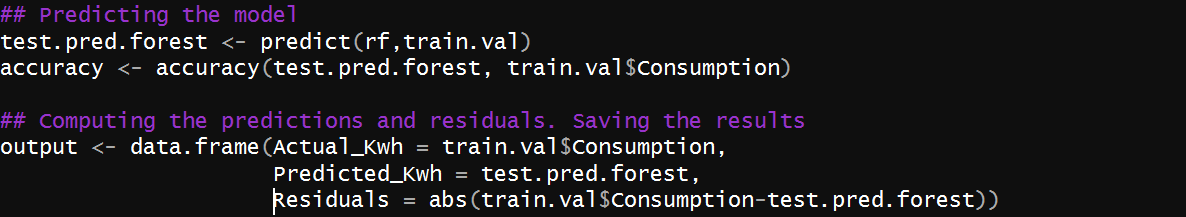
**Note: For the functions and the result, please refer the following files in the Git repository**

1. **Part2/SingleBuilding\_Prediction/** **RandomForest\_Outlier**

We have done outlier detection in below steps.

##### Compute Predictions and Residuals

We predict the values using our Random Forest model. We will then calculate the residuals. Since Random Forest does not give us residuals directly, we can calculate Residuals as shown below, which is nothing but absolute difference of actual verses predicted values.



Difference from ‘Actual\_Kwh’ to ‘Predicted\_Kwh’ gives us ‘Residuals’.

##### Standard Deviation on the Residuals

Standard deviation is calculated using base r function sd().

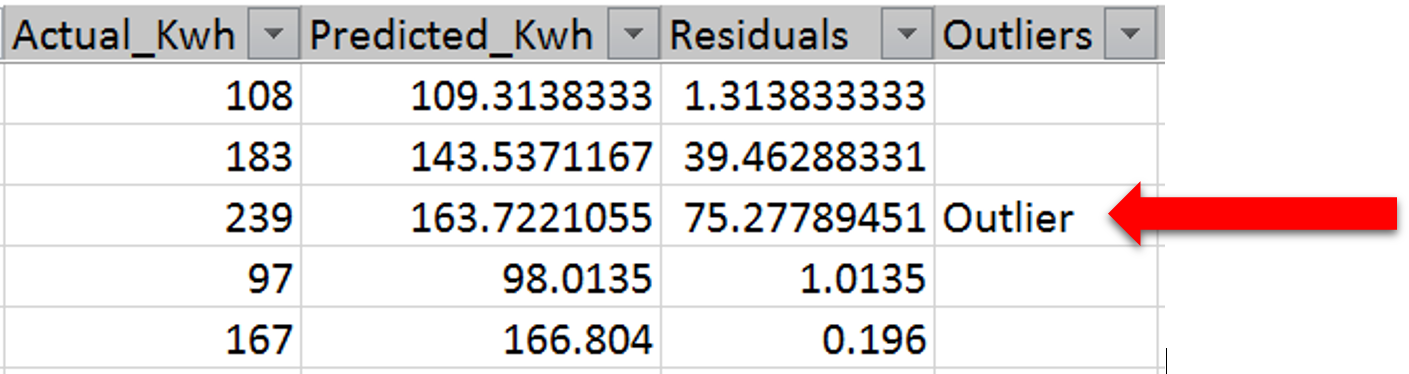


##### Tag points as outliers

We can compute another column called outliers which gives us Outlier if the residual is greater than 2 times the standard deviation



In below example SD was computed as 32.78 for Building 5 unit 5310 meter 1. So any value greater than 65.56 will be marked as an outlier.



From the example we can see that residual 75 < 2\*32.78 hence it is marked as an outlier. Overall outliers detected from all the buildings were around 2%.

## Generic approach: Classifcation

**Note: For the functions and the result, please refer the following files in the Git repository**

1. **Part2/SingleBuilding\_Classifcation/**”RespectiveModel”

Generic steps remain the same, however the feature selection and the formula changes for the model. We will discuss the algorithm used in the below section for all the Machine Learning algorithm used for classifcation

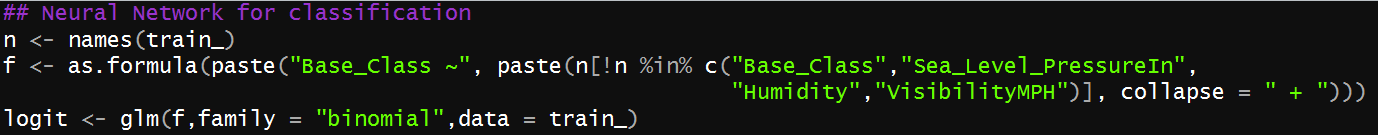
### Different Machine Learning Algorithms and Output

**Note: For the functions and the result, please refer the following files in the Git repository**

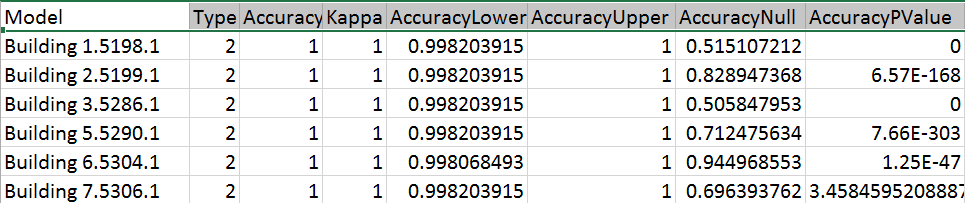
1. **Part2/SingleBuilding\_Classification/**

#### Logistic Regression

**Binary Logistic Regression is a special type of regression where binary response variable is related to a set of explanatory variables, which can be discrete and/or continuous. We ae using glm() function to predict the Base\_Class\_Hour in this case of classification.**

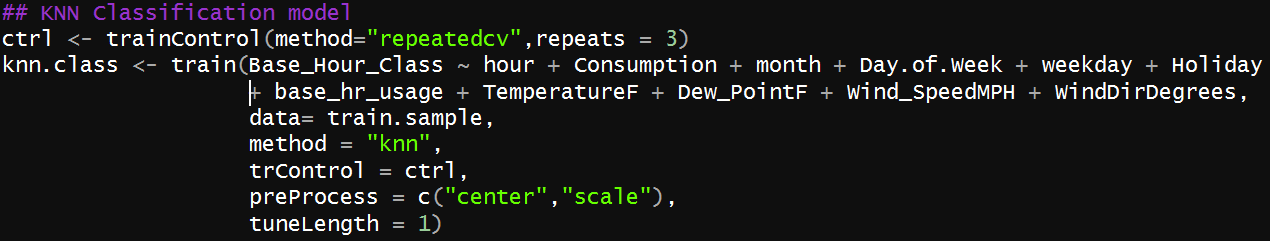


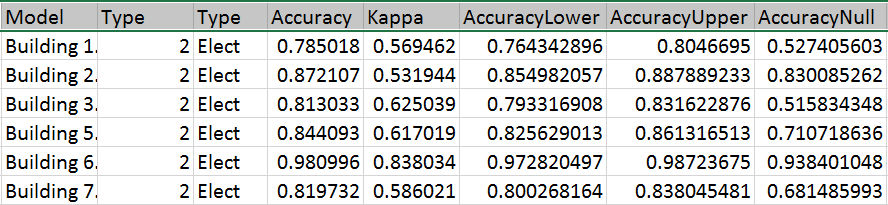
In this case, our function will result the Confusion matrix as shown below and also the ROC curve will be saved as described later



#### KNN

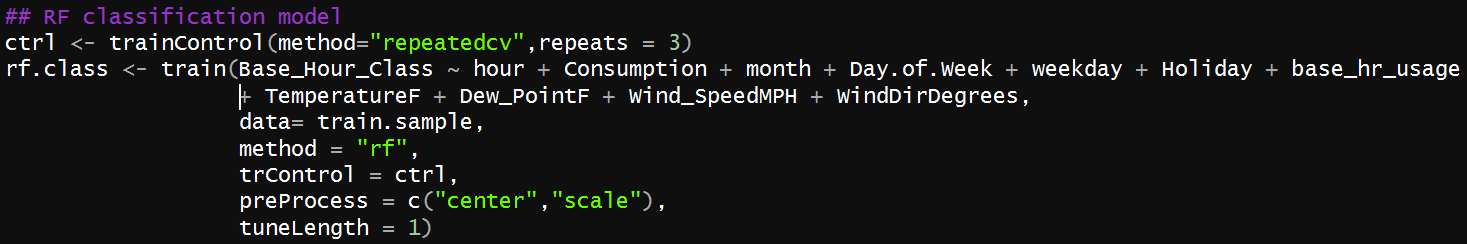
K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). Here we are using caret library with a train control of 3 times repeated cross validation. Model is trained with train control and also a preprocess of center and scale is applied to the features.

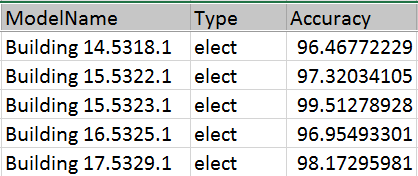




#### Random Forest

Random Forests grows many classification trees. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest). We use caret library to compute the Random Forest similar to what we did earlier in KNN.

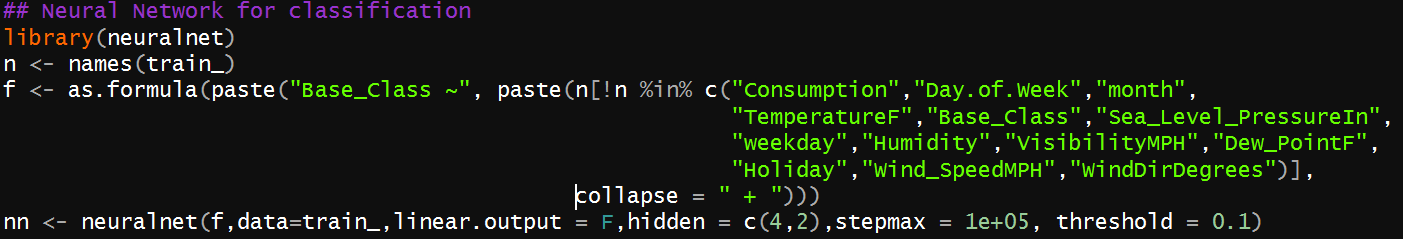


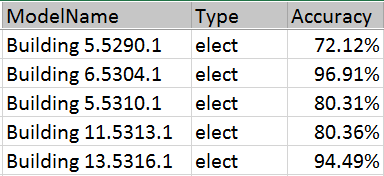


#### Neural Network

Artificial neural networks are relatively crude electronic networks of neurons based on the neural structure of the brain. They process records one at a time, and learn by comparing their classification of the record (i.e., largely arbitrary) with the known actual classification of the record.

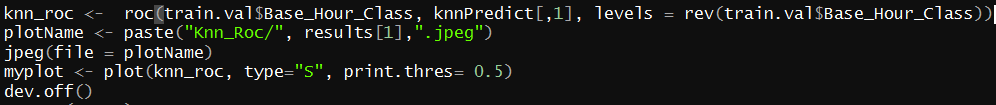
We are using NeuralNet library to train the model with a hidden layer of 3/2 and threshold of 0.1.

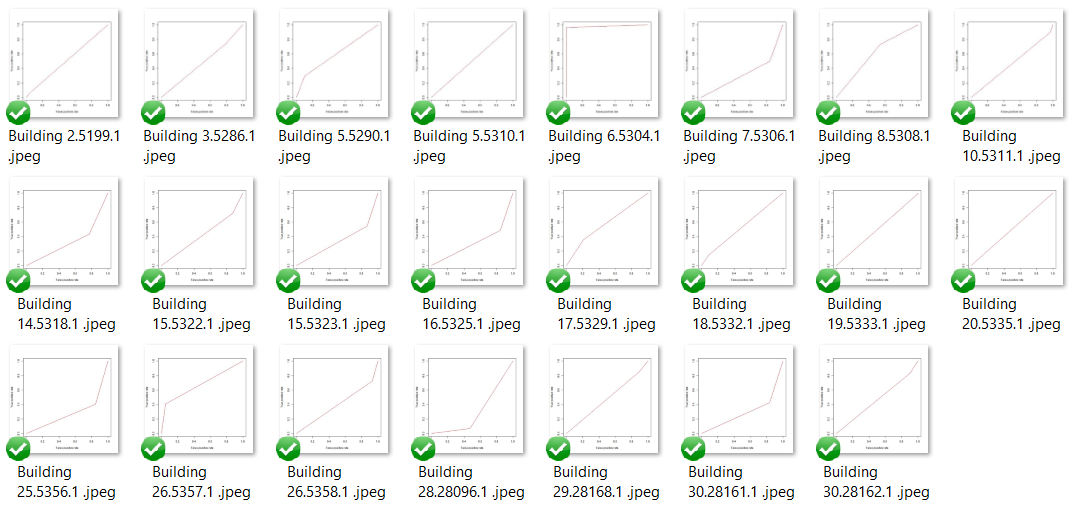




#### ROC charts

We have calculated ROC for all the models. Below is sample code to generate the ROC.



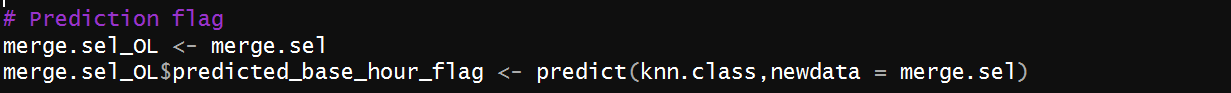


#### Outlier Detection

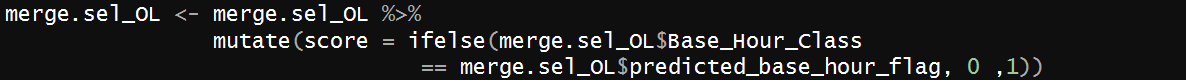
For the best model, we have computed as follows:

##### Computed prediction flag

We have computed the prediction from our dataset so that we can compare it against the actual value.

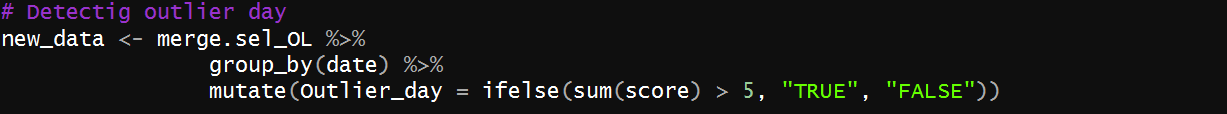


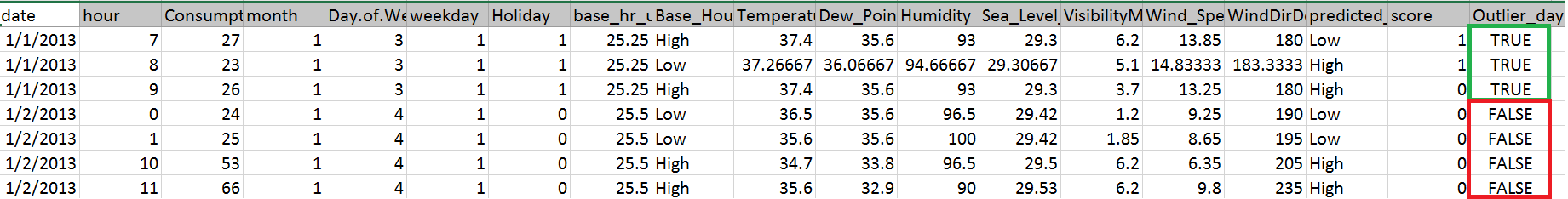
If we have the mismatch between predicted and Actual value we have a new column called ‘Score’ which has value 1 for mismatch and 0 for no mismatch.



##### Outlier\_day

We have computed ‘outlier\_day’ column as If there is a mismatch in flags for 6 or more hours, we tagged all rows for that day as True else False.





## Generic approach: All Buildings (Prediction and Classifiction)

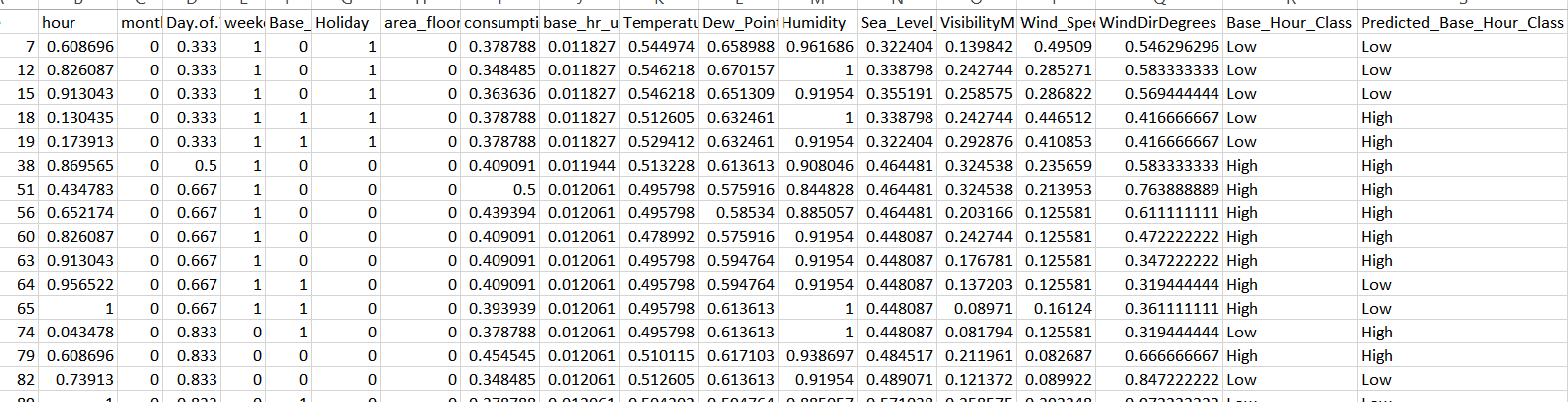
**Note: For the functions and the result, please refer the following files in the Git repository**

1. **Part2/** **AllBuilding\_Script&Result**

For one dataset as a file, we will the follow the same approach as described in part 3.3.2 and 3.3.3.

Code for performing all the machine algorithm on the single dataset is located at **Part2/** **AllBuilding\_Script&Result/** **Part2\_AllBuildings\_AllModel.R**

### Classification output – (KNN Example)



### Prediction output – (KNN Example)

