**FEATURE ENGINEERING:**

Feature engineering is the process of using domain knowledge of the data to create [features](https://en.wikipedia.org/wiki/Feature_(machine_learning)) that make [machine learning](https://en.wikipedia.org/wiki/Machine_learning) algorithms work. Feature engineering is fundamental to the application of machine learning and is both difficult and expensive. The need for manual feature engineering can be obviated by automated [feature learning](https://en.wikipedia.org/wiki/Feature_learning).

In this case we have extracted the day on which the sensor was triggered, the duration for which the occupant stayed in the room, the number of minutes that has passed from the start of the day to when sensor was triggered on.

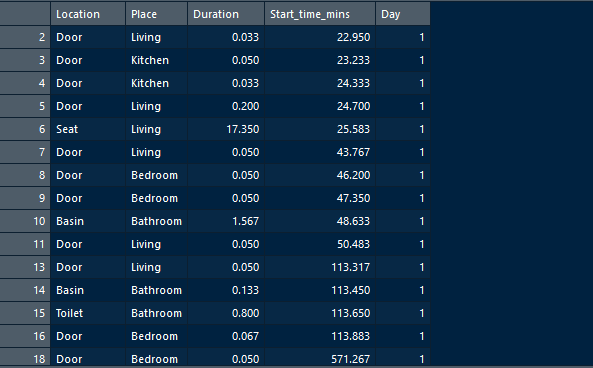


Image of the dataset after Feature Engineering

ALGORITHM

* LINEAR REGRESSION:

Linear regression is a basic and commonly used type of predictive analysis. The overall idea of regression is to examine two things: (1) does a set of predictor variables do a good job in predicting an outcome (dependent) variable? (2) Which variables in particular are significant predictors of the outcome variable, and in what way do they–indicated by the magnitude and sign of the beta estimates–impact the outcome variable? These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables. The simplest form of the regression equation with one dependent and one independent variable is defined by the formula y = c + b\*x, where y = estimated dependent variable score, c = constant, b = regression coefficient, and x = score on the independent variable.

Three major uses for regression analysis are (1) determining the strength of predictors, (2) forecasting an effect, and (3) trend forecasting. Here we are going to use linear regression for the third purpose.

**Multiple linear regression**

1 dependent variable (interval or ratio) , 2+ independent variables (interval or ratio or dichotomous)

**CODE:**

#using the data

dd<- read.csv(file.choose())

dd$Day<-as.numeric(dd$Day)

df<-dd[dd$Duration<30,]

#spliting the data into test and train

require(caTools)

set.seed(101)

sample = sample.split(df, SplitRatio = .75)

train = subset(df, sample == TRUE)

test = subset(df, sample == FALSE)

#dd$Start\_time\_mins<-scale(dd$Start\_time\_mins)

#linear regression model

model\_lr <- lm(Duration~Location+Place+Start\_time\_mins+Day,train )

print(model\_lr)

summary(model\_lr)

#predicting the values for the test data set

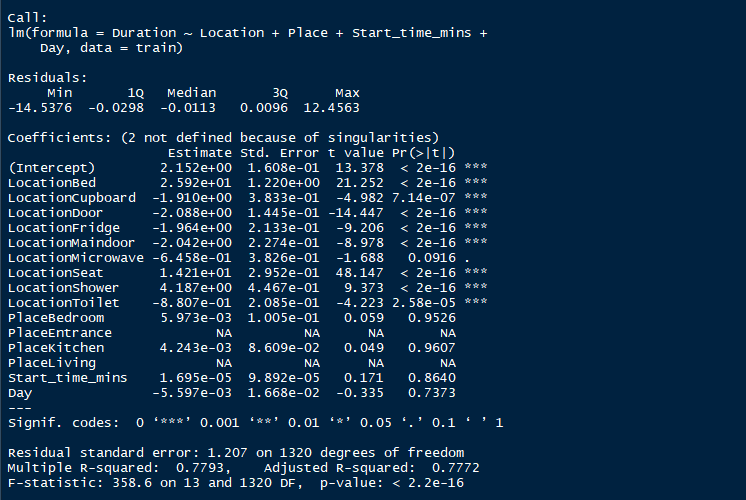
test$pred\_duration<-predict(model\_lr,test)

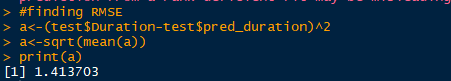
#finding RMSE

a<-(test$Duration-test$pred\_duration)^2

a<-sqrt(mean(a))

print(a)





* REGRESSION TREE

Decision tree learning uses a [decision tree](https://en.wikipedia.org/wiki/Decision_tree) (as a [predictive model](https://en.wikipedia.org/wiki/Predictive_modelling)) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modelling approaches used in [statistics](https://en.wikipedia.org/wiki/Statistics), [data mining](https://en.wikipedia.org/wiki/Data_mining) and [machine learning](https://en.wikipedia.org/wiki/Machine_learning). Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, [leaves](https://en.wikipedia.org/wiki/Leaf_node) represent class labels and branches represent [conjunctions](https://en.wikipedia.org/wiki/Logical_conjunction) of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically [real numbers](https://en.wikipedia.org/wiki/Real_numbers)) are called regression trees.

CODE:

#using the data

dd<- read.csv(file.choose())

df<-dd[dd$Duration<30,]

#spliting the data into test and train

require(caTools)

set.seed(101)

sample = sample.split(df, SplitRatio = .75)

train = subset(df, sample == TRUE)

test = subset(df, sample == FALSE)

#dd$Start\_time\_mins<-scale(dd$Start\_time\_mins)

#regression tree

library(rpart)

library(rpart.plot)

model\_rpart <- rpart(Duration~Location+Place+Start\_time\_mins+Day,train,

control = rpart.control(cp = 0.03979373 ))

#print(model\_rpart)

#summary(model\_rpart)

#predicting the values for the test data set

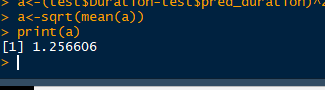
test$pred\_duration<-predict(model\_rpart,test)

#finding RMSE

a<-(test$Duration-test$pred\_duration)^2

a<-sqrt(mean(a))

print(a)



* SUPPORT VECTOR REGRESSION

Support Vector Machine can also be used as a regression method, maintaining all the main features that characterize the algorithm (maximal margin). The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences. First of all, because output is a real number it becomes very difficult to predict the information at hand, which has infinite possibilities. In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem. But besides this fact, there is also a more complicated reason, the algorithm is more complicated therefore to be taken in consideration. However, the main idea is always the same: to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated.

CODE:

#using the data

dd<- read.csv(file.choose())

#validation of the data using kfold

df<-dd[dd$Duration<30,]

print(model)

df<-dd

#df<-dd[dd$Duration<60,]

#spliting the data into test and train

require(caTools)

set.seed(101)

sample = sample.split(df, SplitRatio = .75)

train = subset(df, sample == TRUE)

test = subset(df, sample == FALSE)

#svm

library(e1071)

model\_svm <- svm(Duration~Location+Place+Start\_time\_mins+Day,train)

#predicting the values for the test data set

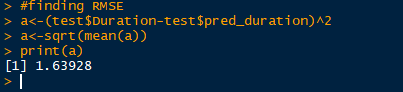
test$pred\_duration<-predict(model\_svm,test)

#finding RMSE

a<-(test$Duration-test$pred\_duration)^2

a<-sqrt(mean(a))

print(a)



* RANDOM FOREST

Random forests or random decision forests ensemble learning method for [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis) and other tasks, that operate by constructing a multitude of [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning) at training time and outputting the class that is the [mode](https://en.wikipedia.org/wiki/Mode_(statistics)) of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of [overfitting](https://en.wikipedia.org/wiki/Overfitting) to their [training set.](https://en.wikipedia.org/wiki/Test_set)

CODE:

#using the data

dd<- read.csv(file.choose())

df<-dd

#df<-dd[dd$Duration<60,]

#spliting the data into test and train

require(caTools)

set.seed(101)

sample = sample.split(df, SplitRatio = .80)

train = subset(df, sample == TRUE)

test = subset(df, sample == FALSE)

#dd$Start\_time\_mins<-scale(dd$Start\_time\_mins)

#RANDOM FOREST

library(randomForest)

library(rpart.plot)

model\_rpart <- randomForest(Duration~.,train,mtry = 3)

print(model\_rpart)

summary(model\_rpart)

#finding RMSE

a<-(test$Duration-test$pred\_duration)^2

a<-sqrt(mean(a))

print(a)

