Project -2

Deepak Kumar

August 8, 2016

We are working on the data collected from different smart watches and data also contain that activities the user was engaged and will built a model to identify the activities of the users in which they are engaged.

In this we will generate two models, one using Logistic Regression and other using Decision Tree

First we import the smart watches data, the data is generated from accelerometer of the smart watch while different user was engaged in different activities. So the data is collected from 9 different users who were engaged in 6 different activities.

The second CSV file contains just accelerometer data and we have to identify the in which activity user was engaged using the model that we generate using the accelerometer data file

## Import Data  
watch\_acc <- read.csv("Watch\_accelerometer.csv")  
HAR<- read.csv("HARnew.csv")

As we want to identify the user activities, and our data also contains some data in which user activities is not identified and it contains null value so here we extracting the data which contains the “null” value in user activity.

Based on the article, it appears that the LG watches have more consistent measurements, so we will just look at the two LG models for this analysis, and also it appears both LG watches has the similar accuracy, so we will not make different predictive model for both different watches. As our data contains the values of two different brands smart watches and we want to use just LG watch, so we extract the subset of LG watch data.

## There are more null vlaues and there is no use of null values in our analysis  
## so we are just removing the null values.  
watch\_acc1 <- subset(watch\_acc, watch\_acc$gt != "null")  
  
## Data Prepration  
watch\_acc1$gt <- factor(watch\_acc1$gt)  
watch\_acc1$out <- relevel(watch\_acc1$gt , ref = "bike")

## Extract The LG Watch Data.  
lgwatch <- subset(watch\_acc1, Model == "lgwatch")  
str(lgwatch)

## 'data.frame': 2655958 obs. of 11 variables:  
## $ Index : int 0 1 2 3 4 5 6 7 8 9 ...  
## $ Arrival\_Time : num 1.42e+12 1.42e+12 1.42e+12 1.42e+12 1.42e+12 ...  
## $ Creation\_Time: num 2.08e+14 2.08e+14 2.08e+14 2.08e+14 2.08e+14 ...  
## $ x : num -9.16 -9.2 -9.21 -9.21 -9.22 ...  
## $ y : num -3.76 -3.79 -3.8 -3.76 -3.74 ...  
## $ z : num 1.4 1.42 1.44 1.45 1.38 ...  
## $ User : Factor w/ 9 levels "a","b","c","d",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ Model : Factor w/ 2 levels "gear","lgwatch": 2 2 2 2 2 2 2 2 2 2 ...  
## $ Device : Factor w/ 4 levels "gear\_1","gear\_2",..: 3 3 3 3 3 3 3 3 3 3 ...  
## $ gt : Factor w/ 6 levels "bike","sit","stairsdown",..: 5 5 5 5 5 5 5 5 5 5 ...  
## $ out : Factor w/ 6 levels "bike","sit","stairsdown",..: 5 5 5 5 5 5 5 5 5 5 ...

As the data is generated through continuous time, instead of randomly dividing the data, into two partition, for this project we will choose randomly 6 of the 9 users in as a training subject and other three as a test subject.

## As the Data is generated through continuous time, Instead of randomly dividing data  
## into two partition, for this project we will choose randomly 6 0f the 9 users(67%)  
## as a training subject and other three as a test subject  
  
user <- unique(lgwatch$User)  
user <- data.frame(user)  
## PArtition the user into training and test set  
pd <- sample(2,nrow(user), replace = TRUE, prob = c(0.67,0.33))  
  
train\_user = user[pd==1,]  
val\_user = user[pd==2,]  
train\_user

## [1] a c d e f g h  
## Levels: a b c d e f g h i

val\_user

## [1] b i  
## Levels: a b c d e f g h i

train <- lgwatch[lgwatch$User %in% train\_user,]  
val <- lgwatch[lgwatch$User %in% val\_user,]

**Logistic Regression Model**

## Logistic Regression Model, package installation  
#install.pacakges("nnet")  
library(nnet)

First we will try to use all variables, as we want to predict the activity of user, so we use the activity predictor as a output variables, as we are not concerned with the user and the model of the device, we will exclude the all that variables, we just include the Arrival time, creation time, three dimensions data of accelerometer (x, y, z).

mylog1 <- multinom(gt~ Arrival\_Time + Creation\_Time +x +y +z , data = train)

## # weights: 42 (30 variable)  
## initial value 3936561.848888   
## final value 3936561.848888   
## converged

sum1 <- summary(mylog1)

## Warning in sqrt(diag(vc)): NaNs produced

print(sum1)

## Call:  
## multinom(formula = gt ~ Arrival\_Time + Creation\_Time + x + y +   
## z, data = train)  
##   
## Coefficients:  
## (Intercept) Arrival\_Time Creation\_Time x  
## sit -2.662462e-30 -3.793294e-18 -4.006799e-16 -2.176607e-30  
## stairsdown -2.202870e-30 -3.138514e-18 -5.238862e-16 -2.380931e-31  
## stairsup -3.744058e-30 -5.334295e-18 -7.463001e-16 1.018179e-29  
## stand -9.454088e-31 -1.347042e-18 -1.674468e-16 1.741845e-29  
## walk 2.276973e-30 3.244214e-18 5.369173e-16 1.676092e-29  
## y z  
## sit -3.588824e-29 1.894229e-29  
## stairsdown 1.906376e-29 -2.146179e-29  
## stairsup 1.799324e-29 -2.279247e-29  
## stand 6.090162e-30 -2.366839e-29  
## walk -9.203588e-30 -2.751802e-29  
##   
## Std. Errors:  
## (Intercept) Arrival\_Time Creation\_Time x  
## sit 4.893460e-26 7.077028e-15 4.055228e-17 1.395730e-25  
## stairsdown 4.998670e-27 7.118227e-15 4.082176e-17 5.628494e-26  
## stairsup 5.050078e-27 7.194314e-15 4.132154e-17 5.699791e-26  
## stand 4.914274e-27 7.000841e-15 4.005613e-17 5.518862e-26  
## walk 4.762609e-27 6.784772e-15 3.866494e-17 5.318344e-26  
## y z  
## sit NaN 1.060136e-26  
## stairsdown 8.632044e-27 8.123375e-27  
## stairsup 8.257898e-27 8.244812e-27  
## stand 9.214004e-27 7.936268e-27  
## walk 1.030752e-26 7.592373e-27  
##   
## Residual Deviance: 7873124   
## AIC: 7873184

cm\_log1 <- table(predict(mylog1), train$gt)  
cm\_log1

##   
## bike sit stairsdown stairsup stand walk  
## bike 0 0 0 0 0 0  
## sit 0 0 0 0 0 0  
## stairsdown 0 0 0 0 0 0  
## stairsup 0 0 0 0 0 0  
## stand 0 0 0 0 0 0  
## walk 495453 318878 327042 299665 349379 406620

1-sum(diag(cm\_log1))/sum(cm\_log1)

## [1] 0.8149235

The above model we generated, it shows the misclassification error rate of 81% and if we look at the confusion matrix, the model didn’t predict any correct activity except the “walk” activity. So we will cancel the model try with by only adding the three dimensions variables of accelerometer, and activity as a output variable.

mylog2 <- multinom(gt ~ x+y+z , data = train)

## # weights: 30 (20 variable)  
## initial value 3936561.848888   
## iter 10 value 3614223.292159  
## iter 20 value 3586667.523707  
## final value 3562687.237848   
## converged

sum <- summary(mylog2)  
print(sum)

## Call:  
## multinom(formula = gt ~ x + y + z, data = train)  
##   
## Coefficients:  
## (Intercept) x y z  
## sit -0.3164691 0.03267522 -0.141378581 -0.1270713  
## stairsdown 0.7854648 0.04591114 -0.004460855 -0.4107398  
## stairsup 0.6876284 0.05446698 -0.010893040 -0.4099195  
## stand 0.8569189 0.06066407 -0.028018298 -0.4461110  
## walk 1.0154590 0.06106266 -0.046348781 -0.4897515  
##   
## Std. Errors:  
## (Intercept) x y z  
## sit 0.004377065 0.0003188512 0.0005463326 0.0008457056  
## stairsdown 0.003565531 0.0003085011 0.0004606174 0.0009175348  
## stairsup 0.003625825 0.0003153146 0.0004746122 0.0009394049  
## stand 0.003539011 0.0003056387 0.0004689539 0.0009175413  
## walk 0.003474380 0.0002977998 0.0004666482 0.0009041259  
##   
## Residual Deviance: 7125374   
## AIC: 7125414

The significance matrix is showing that all variables are significant on all activities. So we will use this model for as our final model

## Check The significance of the variables  
  
z <- sum$coefficients/sum$standard.errors  
p <- (1 - pnorm(abs(z), 0, 1))\*2  
p

## (Intercept) x y z  
## sit 0 0 0 0  
## stairsdown 0 0 0 0  
## stairsup 0 0 0 0  
## stand 0 0 0 0  
## walk 0 0 0 0

Logistic Regression Equation

# Here is the equation for logistic regression

Sit = -0.3164 + 0.0326(x) – 0.14137(y) - 0.127(z)

Stairsdown = 0.7845 + 0.0459(x) – 0.0044(y) - 0.4107(z)

Stairsup = 0.6876 + 0.0544(x) – 0.0108(y) - 0.4099(z)

Stand = 0.856 + 0.0606(x) – 0.0280(y) - 0.446(z)

Walk = 1.0154 + 0.061(x) – 0.046(y) - 0.489(z)

Y1 = ln[p(sit)/p(bike)] = ey1

Y2 = ln[p(stairsdown)/p(bike)] = ey2

Y3 = ln[p(stairsup)/p(bike)] = ey3

Y4 = ln[p(stand)/p(bike)] = ey4

Y5 = ln[p(walk)/p(bike)] = ey1

= ey1 + ey2 + ey3 + ey4 + ey5

P(bike)+p(sit)+p(stairsdown)+p(stairsup)+p(stand)+p(walk) =1

= ey1 + ey2 + ey3 + ey4 + ey5

P(bike) =

**Confusion matrix**

# Confusion Matrix for Training Data set  
  
cm <- table(predict(mylog2), train$gt)  
cm

##   
## bike sit stairsdown stairsup stand walk  
## bike 351941 195664 106175 96210 48797 60735  
## sit 58260 59337 15011 10261 2305 12184  
## stairsdown 12635 1137 20779 12773 13823 14884  
## stairsup 0 0 4 1 0 0  
## stand 199 0 2186 1827 298 6811  
## walk 72418 62740 182887 178593 284156 312006

1-sum(diag(cm))/sum(cm)

## [1] 0.6611973

## Confusion Matrix For Test Data set  
cm1 <- table(predict(mylog2, val), val$gt)  
cm1

##   
## bike sit stairsdown stairsup stand walk  
## bike 64911 21623 19602 15997 16962 14946  
## sit 11947 40072 2999 1198 4226 2899  
## stairsdown 33 0 19 38 2 49  
## stairsup 0 0 0 0 0 0  
## stand 0 0 0 0 0 0  
## walk 10878 3595 64028 67244 44631 51022

1-sum(diag(cm1))/sum(cm1)

## [1] 0.6600199

The confusion matrix is for training data and test data is showing the misclassification error rate is around 66%, which is very high, but it also varying from data to data, as creating the markdown file the code is executing every time and generating the different misclassification error rate. The least misclassification error rate was around 35%for training data and 47% for test data.

Using the above model, we will predict the activity of new data. So in this model we are just using accelerometer three dimensions data, we will extract only the accelerometer dimension variables and name the as x, y, z. and create a new column which contain the activity in which the user was engaged.

#Identify the activities using the Logistic Regression Model to check the individaul activity  
  
HAR <- HAR[,2:4]  
colnames(HAR)<- c("x","y","z")  
HAR$gt\_logistic <- predict(mylog2, HAR)

**Decision Tree**

Now using the same data we will use the decision tree generate a new model and predict the data. First we will try to use the same data variables we use in our final logistic regression model with the 99% probability and we kept the minimum split near to 100000, smaller the minimum split, its giving the high accuracy. But with the smaller minimum split the tree becomes too large, as the following tree has 135 nodes.

## Decision Trees  
  
library(party)

## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

## Loading required package: strucchange

## Loading required package: zoo

##   
## Attaching package: 'zoo'

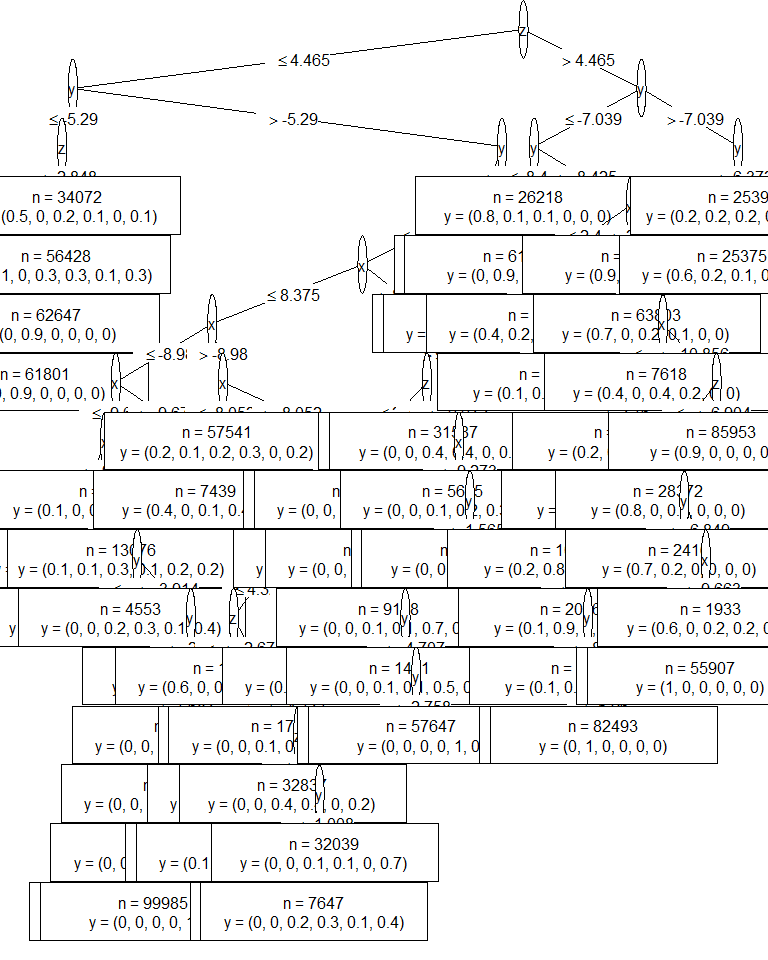
## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

## Loading required package: sandwich

dtree <- ctree(gt ~ x + y + z , data=train, controls = ctree\_control(mincriterion = 0.99, minsplit=100000))  
dtree

##   
## Conditional inference tree with 68 terminal nodes  
##   
## Response: gt   
## Inputs: x, y, z   
## Number of observations: 2197037   
##   
## 1) z <= 4.464767; criterion = 1, statistic = 463155.617  
## 2) y <= -5.290253; criterion = 1, statistic = 64171.837  
## 3) z <= 2.848495; criterion = 1, statistic = 34288.863  
## 4) x <= 3.659897; criterion = 1, statistic = 30670.73  
## 5) z <= 2.26709; criterion = 1, statistic = 25570.817  
## 6) x <= 1.419708; criterion = 1, statistic = 69491.319  
## 7)\* weights = 66979   
## 6) x > 1.419708  
## 8)\* weights = 61801   
## 5) z > 2.26709  
## 9)\* weights = 62647   
## 4) x > 3.659897  
## 10)\* weights = 56428   
## 3) z > 2.848495  
## 11)\* weights = 34072   
## 2) y > -5.290253  
## 12) y <= 8.006943; criterion = 1, statistic = 284560.539  
## 13) y <= 4.631134; criterion = 1, statistic = 133344.089  
## 14) x <= 8.375153; criterion = 1, statistic = 26492.664  
## 15) x <= -8.980133; criterion = 1, statistic = 60849.306  
## 16) x <= -9.670441; criterion = 1, statistic = 101792.122  
## 17) x <= -9.79422; criterion = 1, statistic = 10895.162  
## 18) x <= -16.78749; criterion = 1, statistic = 8948.619  
## 19)\* weights = 7475   
## 18) x > -16.78749  
## 20) y <= -3.654495; criterion = 1, statistic = 6705.271  
## 21)\* weights = 46892   
## 20) y > -3.654495  
## 22)\* weights = 86981   
## 17) x > -9.79422  
## 23)\* weights = 7608   
## 16) x > -9.670441  
## 24) z <= 2.703293; criterion = 1, statistic = 11105.289  
## 25) z <= -0.4655762; criterion = 1, statistic = 13413.758  
## 26)\* weights = 13076   
## 25) z > -0.4655762  
## 27) y <= -3.913956; criterion = 1, statistic = 10239.051  
## 28)\* weights = 4553   
## 27) y > -3.913956  
## 29) y <= -2.943207; criterion = 1, statistic = 14488.581  
## 30) z <= 1.686874; criterion = 1, statistic = 10062.029  
## 31) x <= -9.141739; criterion = 1, statistic = 1893.946  
## 32) z <= 1.489304; criterion = 1, statistic = 1241.707  
## 33) y <= -3.702103; criterion = 1, statistic = 587.364  
## 34)\* weights = 2997   
## 33) y > -3.702103  
## 35)\* weights = 99985   
## 32) z > 1.489304  
## 36)\* weights = 1274   
## 31) x > -9.141739  
## 37)\* weights = 3069   
## 30) z > 1.686874  
## 38)\* weights = 2287   
## 29) y > -2.943207  
## 39)\* weights = 33366   
## 24) z > 2.703293  
## 40)\* weights = 7439   
## 15) x > -8.980133  
## 41) x <= -8.051529; criterion = 1, statistic = 20240.812  
## 42)\* weights = 57541   
## 41) x > -8.051529  
## 43) y <= 4.466888; criterion = 1, statistic = 17699.495  
## 44) x <= 8.156418; criterion = 1, statistic = 16642.586  
## 45) x <= 4.32402; criterion = 1, statistic = 12689.544  
## 46) z <= -2.674561; criterion = 1, statistic = 10590.249  
## 47)\* weights = 17794   
## 46) z > -2.674561  
## 48) z <= 3.022263; criterion = 1, statistic = 28225.222  
## 49) x <= -4.609772; criterion = 1, statistic = 11308.868  
## 50) y <= 1.97226; criterion = 1, statistic = 4838.191  
## 51)\* weights = 97907   
## 50) y > 1.97226  
## 52)\* weights = 14864   
## 49) x > -4.609772  
## 53)\* weights = 36793   
## 48) z > 3.022263  
## 54)\* weights = 33689   
## 45) x > 4.32402  
## 55) z <= 3.469772; criterion = 1, statistic = 21796.565  
## 56) z <= -4.171219; criterion = 1, statistic = 17565.354  
## 57)\* weights = 1701   
## 56) z > -4.171219  
## 58) z <= 0.1872406; criterion = 1, statistic = 11265.214  
## 59)\* weights = 32837   
## 58) z > 0.1872406  
## 60) y <= 1.008209; criterion = 1, statistic = 6485.109  
## 61) x <= 7.965729; criterion = 1, statistic = 4674.848  
## 62)\* weights = 95756   
## 61) x > 7.965729  
## 63)\* weights = 7647   
## 60) y > 1.008209  
## 64)\* weights = 32039   
## 55) z > 3.469772  
## 65)\* weights = 11702   
## 44) x > 8.156418  
## 66)\* weights = 17156   
## 43) y > 4.466888  
## 67)\* weights = 9355   
## 14) x > 8.375153  
## 68) x <= 9.172836; criterion = 1, statistic = 124056.791  
## 69) z <= 2.181992; criterion = 1, statistic = 14207.414  
## 70) z <= 0.5151367; criterion = 1, statistic = 55424.988  
## 71)\* weights = 16579   
## 70) z > 0.5151367  
## 72) z <= 1.236389; criterion = 1, statistic = 10184.842  
## 73)\* weights = 29838   
## 72) z > 1.236389  
## 74) x <= 8.556061; criterion = 1, statistic = 4583.608  
## 75)\* weights = 9188   
## 74) x > 8.556061  
## 76) y <= -4.707062; criterion = 1, statistic = 1281.289  
## 77)\* weights = 1451   
## 76) y > -4.707062  
## 78) y <= 2.757782; criterion = 1, statistic = 751.568  
## 79)\* weights = 84576   
## 78) y > 2.757782  
## 80)\* weights = 57647   
## 69) z > 2.181992  
## 81)\* weights = 18434   
## 68) x > 9.172836  
## 82) z <= -0.07221985; criterion = 1, statistic = 12986.745  
## 83)\* weights = 31587   
## 82) z > -0.07221985  
## 84) x <= 9.272812; criterion = 1, statistic = 5595.587  
## 85)\* weights = 5675   
## 84) x > 9.272812  
## 86) y <= 1.565216; criterion = 1, statistic = 4966.454  
## 87)\* weights = 97462   
## 86) y > 1.565216  
## 88)\* weights = 29079   
## 13) y > 4.631134  
## 89) x <= -2.22464; criterion = 1, statistic = 31881.057  
## 90)\* weights = 75232   
## 89) x > -2.22464  
## 91)\* weights = 30494   
## 12) y > 8.006943  
## 92) z <= 3.573914; criterion = 1, statistic = 52683.81  
## 93)\* weights = 49243   
## 92) z > 3.573914  
## 94)\* weights = 61434   
## 1) z > 4.464767  
## 95) y <= -7.039383; criterion = 1, statistic = 111326.467  
## 96) y <= -8.424759; criterion = 1, statistic = 44943.184  
## 97)\* weights = 26218   
## 96) y > -8.424759  
## 98) x <= 2.414963; criterion = 1, statistic = 11400.584  
## 99) x <= -0.8794708; criterion = 1, statistic = 74480.17  
## 100)\* weights = 8566   
## 99) x > -0.8794708  
## 101) x <= 1.153366; criterion = 1, statistic = 2693.843  
## 102)\* weights = 8666   
## 101) x > 1.153366  
## 103) x <= 2.160263; criterion = 1, statistic = 699.728  
## 104) z <= 6.65947; criterion = 1, statistic = 221.075  
## 105) z <= 5.15686; criterion = 1, statistic = 3026.853  
## 106)\* weights = 1070   
## 105) z > 5.15686  
## 107) z <= 5.455002; criterion = 1, statistic = 537.803  
## 108)\* weights = 2026   
## 107) z > 5.455002  
## 109) y <= -8.180023; criterion = 1, statistic = 174.907  
## 110)\* weights = 1088   
## 109) y > -8.180023  
## 111) z <= 5.95964; criterion = 1, statistic = 136.397  
## 112)\* weights = 19773   
## 111) z > 5.95964  
## 113)\* weights = 82493   
## 104) z > 6.65947  
## 114)\* weights = 1094   
## 103) x > 2.160263  
## 115)\* weights = 2233   
## 98) x > 2.414963  
## 116)\* weights = 12964   
## 95) y > -7.039383  
## 117) y <= 6.371628; criterion = 1, statistic = 36128.294  
## 118) z <= 8.980331; criterion = 1, statistic = 17826.82  
## 119) z <= 5.274094; criterion = 1, statistic = 18393.968  
## 120)\* weights = 63803   
## 119) z > 5.274094  
## 121) x <= -10.85561; criterion = 1, statistic = 8457.576  
## 122)\* weights = 7618   
## 121) x > -10.85561  
## 123) z <= 6.904053; criterion = 1, statistic = 5154.21  
## 124) z <= 5.587707; criterion = 1, statistic = 2270.729  
## 125)\* weights = 28372   
## 124) z > 5.587707  
## 126) y <= -6.849396; criterion = 1, statistic = 1503.388  
## 127)\* weights = 2416   
## 126) y > -6.849396  
## 128) x <= 9.662933; criterion = 1, statistic = 1313.735  
## 129) x <= 2.833908; criterion = 1, statistic = 2059.832  
## 130)\* weights = 70447   
## 129) x > 2.833908  
## 131)\* weights = 55907   
## 128) x > 9.662933  
## 132)\* weights = 1933   
## 123) z > 6.904053  
## 133)\* weights = 85953   
## 118) z > 8.980331  
## 134)\* weights = 25375   
## 117) y > 6.371628  
## 135)\* weights = 25393

#plot(dtree)  
plot(dtree ,inner\_panel = node\_inner(dtree, abbreviate = FALSE, pval = FALSE, id=FALSE), terminal\_panel = node\_terminal(dtree, abbreviate = TRUE, digits = 1, fill = c("white"), id= FALSE), cex = 0.8, type = "simple")



# Confusion Matrix for Traing Data set  
  
tab <- table(predict(dtree), train$gt)  
tab

##   
## bike sit stairsdown stairsup stand walk  
## bike 410739 14200 64815 32128 12704 27475  
## sit 6556 286729 4484 3635 896 2025  
## stairsdown 21163 1627 59326 31040 5980 17945  
## stairsup 28847 7562 68152 93842 12605 57244  
## stand 3144 1492 6864 10159 297155 15633  
## walk 25004 7268 123401 128861 20039 286298

1-sum(diag(tab))/sum(tab)

## [1] 0.3472622

## Confusion matrix for Test Data set  
  
tab1 <- table(predict(dtree, val), val$gt)  
tab1

##   
## bike sit stairsdown stairsup stand walk  
## bike 67432 49133 5586 5222 5388 2176  
## sit 2798 12527 1170 1374 8974 899  
## stairsdown 529 7 4318 3135 3008 1433  
## stairsup 3504 682 22171 28590 14791 25607  
## stand 273 1849 8941 8212 20331 2683  
## walk 13233 1092 44462 37944 13329 36118

1-sum(diag(tab1))/sum(tab1)

## [1] 0.6310563

The confusion matrix is for training data is showing the misclassification error rate is around 35%, and for the test data it is around 63% which is very high, but it also varying from data to data. We will use this as a final model. I also generated the model using the all variables in data, but the previous model is more accurate than this, the misclassification error rate of training data was about 40% and for test data, it was same 63%.

Using the above model, we will predict the activity of new data. So in this model we are just using accelerometer three dimensions, we will extract only the accelerometer dimension variables and name the as x, y, z. and create a new column which contain the activity in which the user was engaged.

## Identify the activities using the Decision Tree Model to check the individaul activity  
HAR$tree\_gt <- predict(dtree, HAR)

Generating the output and creating the csv file for the output, the 4th column shows the values we predicted using the logistic regression model and 5th column shows the value we predicted using the decision tree.

head(HAR)

## x y z gt\_logistic tree\_gt  
## 1 9.167398 3.272275 1.729811 walk stand  
## 2 9.152434 3.190872 1.799841 walk stand  
## 3 8.875904 3.444059 3.013103 bike walk  
## 4 8.975264 2.868852 2.032677 walk stand  
## 5 9.059659 2.829348 2.005144 walk stand  
## 6 9.267955 2.238579 1.091157 walk walk

write.csv(HAR, "HAR\_MODEL.csv")

Conclusion: The decision tree model seems to be more accurate than Logistic Regression Model. But both model misclassification error rate is too high. So the accelerometer prediction is not seems more accurate as the misclassification error is showing.