**Project Code:**

########################################

#perform hold-out cross validation

########################################

###getwd() will give current working directory###

###read csv file###

filename = paste("~/vyshya/IUPUI/mybooks/fall2015/intelligent\_systems/prject/data/training/","training.csv",sep="")

print(filename)

complete\_data <- read.csv(filename, stringsAsFactors=F)

str(complete\_data)

#install foreach.zip and iterators.zip.Install these and execute:

library(foreach)

library(entropy)

images <- foreach(i = complete\_data$Image, .combine=rbind) %dopar% {

as.integer(unlist(strsplit(i, " ")))

}

head(images)

#####get 80% of the data using permutation#####

set.seed(0)

perm <- sample(nrow(complete\_data), nrow(complete\_data)\*0.8)

#####split into 80% training data and 20% test data#####

training\_data <- complete\_data[perm,]

test\_data <- complete\_data[-perm,]

training\_images <- images[perm,]

test\_images <- images[-perm,]

head(training\_images)

head(test\_images)

###########################################################

#perform histogram stretching for training images

###########################################################

l = 0

u = 255

stretched\_training\_images = matrix(0,5639,9216)

#row index

i=1

while(i<=5639){

a = 255

b = 0

j=1

while(j<=9216){

p = training\_images[i,j]

if(isTRUE(p<a)){a=p}

if(isTRUE(p>b)){b=p}

j = j+1

}

a = a \* 0.05

b = b \* 0.95

firstPart = (u-l)/(b-a)

j = 1

while(j<=9216){

secondPart = training\_images[i,j] - a

stretched\_training\_images[i,j] = floor((firstPart\*secondPart) + l)

j = j + 1

}

i = i+1

}

head(stretched\_training\_images)

dim(stretched\_training\_images)

###########################################################

#perform histogram stretching for test images

###########################################################

l = 0

u = 255

stretched\_test\_images = matrix(0,1410,9216)

#row index

i=1

while(i<=1410){

a = 255

b = 0

j=1

while(j<=9216){

p = test\_images[i,j]

if(isTRUE(p<a)){a=p}

if(isTRUE(p>b)){b=p}

j = j+1

}

a = a \* 0.05

b = b \* 0.95

firstPart = (u-l)/(b-a)

j = 1

while(j<=9216){

secondPart = test\_images[i,j] - a

stretched\_test\_images[i,j] = floor((firstPart\*secondPart) + l)

j = j + 1

}

i = i+1

}

head(stretched\_test\_images)

dim(stretched\_test\_images)

#####################################################

#display normal and stretched training image

#####################################################

#normal

im <- matrix(data=rev(training\_images[1,]), nrow=96, ncol=96)

image(1:96, 1:96, im, col=gray((0:255)/255))

#stretched

im <- matrix(data=rev(stretched\_training\_images[1,]), nrow=96, ncol=96)

image(1:96, 1:96, im, col=gray((0:255)/255))

#####################################################

#PCA

#####################################################

pca\_train<-prcomp(stretched\_training\_images, center = TRUE,scale. = TRUE)

restored\_matrix\_train <- pca\_train$x[,1:125] %\*% t(pca\_train$rotation[,1:125])

#Display PCA Image

rst <- matrix(data=rev(restored\_matrix\_train[1,]), nrow=96, ncol=96)

image(1:96, 1:96, rst, col=gray((0:255)/255))

#######################################################

#Mean Patch Search

#######################################################

################################Left Eye Center###############################################

################################################################

#Determination of patch\_size and mean\_patch for left eye center

################################################################

coord\_left\_eye <- "left\_eye\_center"

patch\_size <- 10

coord\_lefteyecenter\_x <- paste(coord\_left\_eye, "x", sep="\_")

coord\_lefteyecenter\_y <- paste(coord\_left\_eye, "y", sep="\_")

patches <- foreach (i = 1:nrow(training\_data), .combine=rbind) %dopar% {

if(i<=125){

temp\_matrix <- matrix(data = restored\_matrix\_train[i,], nrow=96, ncol=96)

x <- training\_data[i, coord\_lefteyecenter\_x]

y <- training\_data[i, coord\_lefteyecenter\_y]

pix\_left <- (x-patch\_size)

pix\_right <- (x+patch\_size)

pix\_bottom <- (y-patch\_size)

pix\_top <- (y+patch\_size)

if ( (!is.na(x)) && (!is.na(y)) && (pix\_left>=1) && (pix\_right<=96) && (pix\_bottom>=1) && (pix\_top<=96) )

{

as.vector(temp\_matrix[pix\_left:pix\_right, pix\_bottom:pix\_top])

}

else

{

NULL

}

}

}

mean.patch.lefteyecenter <- matrix(data = colMeans(patches), nrow=2\*patch\_size+1, ncol=2\*patch\_size+1)

###########################################################

#Plot and display the mean patch image of left eye

###########################################################

image(1:21, 1:21, mean.patch.lefteyecenter[21:1,21:1], col=gray((0:255)/255))

###########################################################

#Search\_size determination for left eye center

###########################################################

mean.patchInput.data = training\_data[1:125,]

dim(mean.patchInput.data)

output <- foreach(k =1:5, .combine=rbind) %do%{

search\_size <- k

mean\_lefteyecenter\_x <- mean(mean.patchInput.data[, coord\_lefteyecenter\_x], na.rm=T)

mean\_lefteyecenter\_y <- mean(mean.patchInput.data[, coord\_lefteyecenter\_y], na.rm=T)

test.pix.left <- as.integer(mean\_lefteyecenter\_x)-search\_size

test.pix.right <- as.integer(mean\_lefteyecenter\_x)+search\_size

test.pix.bottom <- as.integer(mean\_lefteyecenter\_y)-search\_size

test.pix.top <- as.integer(mean\_lefteyecenter\_y)+search\_size

candidate\_pixels\_lefteyecenter <- expand.grid(x = test.pix.left:test.pix.right, y = test.pix.bottom:test.pix.top)

#################################################

#Correlation check with a given test image

#################################################

test\_image\_1 <- matrix(data = test\_images[1,], nrow=96, ncol=96)

correlation\_values\_lefteyecenter <- foreach(j = 1:nrow(candidate\_pixels\_lefteyecenter), .combine=rbind) %dopar% {

x <- candidate\_pixels\_lefteyecenter$x[j]

y <- candidate\_pixels\_lefteyecenter$y[j]

candidate\_patch <- test\_image\_1[(x-patch\_size):(x+patch\_size), (y-patch\_size):(y+patch\_size)]

score <- cor(as.vector(candidate\_patch), as.vector(mean.patch.lefteyecenter))

score <- ifelse(is.na(score), 0, score)

data.frame(x, y, score)

}

correlation\_values\_lefteyecenter

candidate\_pixel\_and\_score\_lefteyecenter<- correlation\_values\_lefteyecenter[which.max(correlation\_values\_lefteyecenter$score), c("x", "y")]

candidate\_pixel\_and\_score\_lefteyecenter

################################################################################

#Calculate training error(MSE) for left eye center using MSE formula

################################################################################

left\_eye\_center\_x\_trainingvalues = data.frame(training\_data[,coord\_lefteyecenter\_x])

left\_eye\_center\_y\_trainingvalues = data.frame(training\_data[,coord\_lefteyecenter\_y])

# calculate training error for left\_eye\_center\_x

left\_eye\_center\_x\_trainingerror\_MSE = 0

count = 0

foreach(i = 1:nrow(left\_eye\_center\_x\_trainingvalues)) %do% {

if(!is.na(left\_eye\_center\_x\_trainingvalues[i,1])){

left\_eye\_center\_x\_trainingerror\_MSE = as.numeric(left\_eye\_center\_x\_trainingerror\_MSE) + (as.numeric(candidate\_pixel\_and\_score\_lefteyecenter$x) - as.numeric(left\_eye\_center\_x\_trainingvalues[i,1]))\*(as.numeric(candidate\_pixel\_and\_score\_lefteyecenter$x) - as.numeric(left\_eye\_center\_x\_trainingvalues[i,1]))

count = count + 1

}

}

left\_eye\_center\_x\_trainingerror = as.numeric(left\_eye\_center\_x\_trainingerror\_MSE/count)

left\_eye\_center\_x\_trainingerror = sqrt(left\_eye\_center\_x\_trainingerror)

left\_eye\_center\_x\_trainingerror

# calculate training error for left\_eye\_center\_y

count =0

left\_eye\_center\_y\_trainingerror\_MSE = 0

foreach(i = 1:nrow(left\_eye\_center\_y\_trainingvalues)) %do% {

if(!is.na(left\_eye\_center\_y\_trainingvalues[i,1])){

left\_eye\_center\_y\_trainingerror\_MSE = as.numeric(left\_eye\_center\_y\_trainingerror\_MSE) + (as.numeric(candidate\_pixel\_and\_score\_lefteyecenter$y) - as.numeric(left\_eye\_center\_y\_trainingvalues[i,1]))\*(as.numeric(candidate\_pixel\_and\_score\_lefteyecenter$y) - as.numeric(left\_eye\_center\_y\_trainingvalues[i,1]))

count = count+1

}

}

left\_eye\_center\_y\_trainingerror = as.numeric(left\_eye\_center\_y\_trainingerror\_MSE/count)

left\_eye\_center\_y\_trainingerror = sqrt(left\_eye\_center\_y\_trainingerror)

left\_eye\_center\_y\_trainingerror

#calculate total training error for left\_eye\_center

left\_eye\_center\_trainingerror = as.numeric(((left\_eye\_center\_x\_trainingerror^2)+ (left\_eye\_center\_y\_trainingerror^2))/2)

left\_eye\_center\_trainingerror = sqrt(left\_eye\_center\_trainingerror)

left\_eye\_center\_trainingerror

################################################################################

#Calculate test error(MSE) for left eye center using MSE formula

################################################################################

left\_eye\_center\_x\_testvalues = data.frame(test\_data[,coord\_lefteyecenter\_x])

left\_eye\_center\_y\_testvalues = data.frame(test\_data[,coord\_lefteyecenter\_y])

# calculate test error for left\_eye\_center\_x

left\_eye\_center\_x\_testerror\_MSE = 0

count = 0

foreach(i = 1:nrow(left\_eye\_center\_x\_testvalues)) %do% {

if(!is.na(left\_eye\_center\_x\_testvalues[i,1])){

left\_eye\_center\_x\_testerror\_MSE = as.numeric(left\_eye\_center\_x\_testerror\_MSE) + (as.numeric(candidate\_pixel\_and\_score\_lefteyecenter$x) - as.numeric(left\_eye\_center\_x\_testvalues[i,1]))\*(as.numeric(candidate\_pixel\_and\_score\_lefteyecenter$x) - as.numeric(left\_eye\_center\_x\_testvalues[i,1]))

count = count + 1

}

}

left\_eye\_center\_x\_testerror = as.numeric(left\_eye\_center\_x\_testerror\_MSE/count)

left\_eye\_center\_x\_testerror = sqrt(left\_eye\_center\_x\_testerror)

left\_eye\_center\_x\_testerror

# calculate test error for left\_eye\_center\_y

left\_eye\_center\_y\_testerror\_MSE = 0

count = 0

foreach(i = 1:nrow(left\_eye\_center\_y\_testvalues)) %do% {

if(!is.na(left\_eye\_center\_y\_testvalues[i,1])){

left\_eye\_center\_y\_testerror\_MSE = as.numeric(left\_eye\_center\_y\_testerror\_MSE) + (as.numeric(candidate\_pixel\_and\_score\_lefteyecenter$y) - as.numeric(left\_eye\_center\_y\_testvalues[i,1]))\*(as.numeric(candidate\_pixel\_and\_score\_lefteyecenter$y) - as.numeric(left\_eye\_center\_y\_testvalues[i,1]))

count = count + 1

}

}

left\_eye\_center\_y\_testerror = as.numeric(left\_eye\_center\_y\_testerror\_MSE/count)

left\_eye\_center\_y\_testerror = sqrt(left\_eye\_center\_y\_testerror)

left\_eye\_center\_y\_testerror

#calculate total test error for left\_eye\_center

left\_eye\_center\_testerror = as.numeric(((left\_eye\_center\_x\_testerror^2)+(left\_eye\_center\_y\_testerror^2))/2)

left\_eye\_center\_testerror = sqrt(left\_eye\_center\_testerror)

left\_eye\_center\_testerror

lec\_x <- candidate\_pixel\_and\_score\_lefteyecenter$x

lec\_y <- candidate\_pixel\_and\_score\_lefteyecenter$y

lec\_trainingerror <- left\_eye\_center\_trainingerror

lec\_testerror <- left\_eye\_center\_testerror

data.frame(search\_size,lec\_x, lec\_y, lec\_trainingerror, lec\_testerror)

}

#print the training and test errors for left eye center to an output file

write.csv(output,"~/vyshya/IUPUI/mybooks/fall2015/intelligent\_systems/prject/data/outputErrors\_lefteyecenter.csv")

########################################Left Eye Center##########################################

################################Right Eye Center###############################################

################################################################

#Determination of patch\_size and mean\_patch for right eye center

################################################################

coord\_right\_eye <- "right\_eye\_center"

patch\_size <- 10

coord\_righteyecenter\_x <- paste(coord\_right\_eye, "x", sep="\_")

coord\_righteyecenter\_y <- paste(coord\_right\_eye, "y", sep="\_")

patches <- foreach (i = 1:nrow(training\_data), .combine=rbind) %dopar% {

if(i<=125){

temp\_matrix <- matrix(data = restored\_matrix\_train[i,], nrow=96, ncol=96)

x <- training\_data[i, coord\_righteyecenter\_x]

y <- training\_data[i, coord\_righteyecenter\_y]

pix\_left <- (x-patch\_size)

pix\_right <- (x+patch\_size)

pix\_bottom <- (y-patch\_size)

pix\_top <- (y+patch\_size)

if ( (!is.na(x)) && (!is.na(y)) && (pix\_left>=1) && (pix\_right<=96) && (pix\_bottom>=1) && (pix\_top<=96) )

{

as.vector(temp\_matrix[pix\_left:pix\_right, pix\_bottom:pix\_top])

}

else

{

NULL

}

}

}

mean.patch.righteyecenter <- matrix(data = colMeans(patches), nrow=2\*patch\_size+1, ncol=2\*patch\_size+1)

###########################################################

#Plot and display the mean patch image of right eye

###########################################################

image(1:21, 1:21, mean.patch.righteyecenter[21:1,21:1], col=gray((0:255)/255))

###########################################################

#Search\_size determination for right eye center

###########################################################

mean.patchInput.data = training\_data[1:125,]

dim(mean.patchInput.data)

output <- foreach(k =1:5, .combine=rbind) %do%{

search\_size <- k

mean\_righteyecenter\_x <- mean(mean.patchInput.data[, coord\_righteyecenter\_x], na.rm=T)

mean\_righteyecenter\_y <- mean(mean.patchInput.data[, coord\_righteyecenter\_y], na.rm=T)

test.pix.left <- as.integer(mean\_righteyecenter\_x)-search\_size

test.pix.right <- as.integer(mean\_righteyecenter\_x)+search\_size

test.pix.bottom <- as.integer(mean\_righteyecenter\_y)-search\_size

test.pix.top <- as.integer(mean\_righteyecenter\_y)+search\_size

candidate\_pixels\_righteyecenter <- expand.grid(x = test.pix.left:test.pix.right, y = test.pix.bottom:test.pix.top)

#################################################

#Correlation check with a given test image

#################################################

test\_image\_1 <- matrix(data = test\_images[1,], nrow=96, ncol=96)

correlation\_values\_righteyecenter <- foreach(j = 1:nrow(candidate\_pixels\_righteyecenter), .combine=rbind) %dopar% {

x <- candidate\_pixels\_righteyecenter$x[j]

y <- candidate\_pixels\_righteyecenter$y[j]

candidate\_patch <- test\_image\_1[(x-patch\_size):(x+patch\_size), (y-patch\_size):(y+patch\_size)]

score <- cor(as.vector(candidate\_patch), as.vector(mean.patch.righteyecenter))

score <- ifelse(is.na(score), 0, score)

data.frame(x, y, score)

}

correlation\_values\_righteyecenter

candidate\_pixel\_and\_score\_righteyecenter<- correlation\_values\_righteyecenter[which.max(correlation\_values\_righteyecenter$score), c("x", "y")]

candidate\_pixel\_and\_score\_righteyecenter

################################################################################

#Calculate training error(MSE) for right eye center using MSE formula

################################################################################

right\_eye\_center\_x\_trainingvalues = data.frame(training\_data[,coord\_righteyecenter\_x])

right\_eye\_center\_y\_trainingvalues = data.frame(training\_data[,coord\_righteyecenter\_y])

# calculate training error for right\_eye\_center\_x

right\_eye\_center\_x\_trainingerror\_MSE = 0

count = 0

foreach(i = 1:nrow(right\_eye\_center\_x\_trainingvalues)) %do% {

if(!is.na(right\_eye\_center\_x\_trainingvalues[i,1])){

right\_eye\_center\_x\_trainingerror\_MSE = as.numeric(right\_eye\_center\_x\_trainingerror\_MSE) + (as.numeric(candidate\_pixel\_and\_score\_righteyecenter$x) - as.numeric(right\_eye\_center\_x\_trainingvalues[i,1]))\*(as.numeric(candidate\_pixel\_and\_score\_righteyecenter$x) - as.numeric(right\_eye\_center\_x\_trainingvalues[i,1]))

count = count + 1

}

}

right\_eye\_center\_x\_trainingerror = as.numeric(right\_eye\_center\_x\_trainingerror\_MSE/count)

right\_eye\_center\_x\_trainingerror = sqrt(right\_eye\_center\_x\_trainingerror)

right\_eye\_center\_x\_trainingerror

# calculate training error for right\_eye\_center\_y

count =0

right\_eye\_center\_y\_trainingerror\_MSE = 0

foreach(i = 1:nrow(right\_eye\_center\_y\_trainingvalues)) %do% {

if(!is.na(right\_eye\_center\_y\_trainingvalues[i,1])){

right\_eye\_center\_y\_trainingerror\_MSE = as.numeric(right\_eye\_center\_y\_trainingerror\_MSE) + (as.numeric(candidate\_pixel\_and\_score\_righteyecenter$y) - as.numeric(right\_eye\_center\_y\_trainingvalues[i,1]))\*(as.numeric(candidate\_pixel\_and\_score\_righteyecenter$y) - as.numeric(right\_eye\_center\_y\_trainingvalues[i,1]))

count = count+1

}

}

right\_eye\_center\_y\_trainingerror = as.numeric(right\_eye\_center\_y\_trainingerror\_MSE/count)

right\_eye\_center\_y\_trainingerror = sqrt(right\_eye\_center\_y\_trainingerror)

right\_eye\_center\_y\_trainingerror

#calculate total training error for right\_eye\_center

right\_eye\_center\_trainingerror = as.numeric(((right\_eye\_center\_x\_trainingerror^2)+ (right\_eye\_center\_y\_trainingerror^2))/2)

right\_eye\_center\_trainingerror = sqrt(right\_eye\_center\_trainingerror)

right\_eye\_center\_trainingerror

################################################################################

#Calculate test error(MSE) for right eye center using MSE formula

################################################################################

right\_eye\_center\_x\_testvalues = data.frame(test\_data[,coord\_righteyecenter\_x])

right\_eye\_center\_y\_testvalues = data.frame(test\_data[,coord\_righteyecenter\_y])

# calculate test error for right\_eye\_center\_x

right\_eye\_center\_x\_testerror\_MSE = 0

count = 0

foreach(i = 1:nrow(right\_eye\_center\_x\_testvalues)) %do% {

if(!is.na(right\_eye\_center\_x\_testvalues[i,1])){

right\_eye\_center\_x\_testerror\_MSE = as.numeric(right\_eye\_center\_x\_testerror\_MSE) + (as.numeric(candidate\_pixel\_and\_score\_righteyecenter$x) - as.numeric(right\_eye\_center\_x\_testvalues[i,1]))\*(as.numeric(candidate\_pixel\_and\_score\_righteyecenter$x) - as.numeric(right\_eye\_center\_x\_testvalues[i,1]))

count = count + 1

}

}

right\_eye\_center\_x\_testerror = as.numeric(right\_eye\_center\_x\_testerror\_MSE/count)

right\_eye\_center\_x\_testerror = sqrt(right\_eye\_center\_x\_testerror)

right\_eye\_center\_x\_testerror

# calculate test error for right\_eye\_center\_y

right\_eye\_center\_y\_testerror\_MSE = 0

count = 0

foreach(i = 1:nrow(right\_eye\_center\_y\_testvalues)) %do% {

if(!is.na(right\_eye\_center\_y\_testvalues[i,1])){

right\_eye\_center\_y\_testerror\_MSE = as.numeric(right\_eye\_center\_y\_testerror\_MSE) + (as.numeric(candidate\_pixel\_and\_score\_righteyecenter$y) - as.numeric(right\_eye\_center\_y\_testvalues[i,1]))\*(as.numeric(candidate\_pixel\_and\_score\_righteyecenter$y) - as.numeric(right\_eye\_center\_y\_testvalues[i,1]))

count = count + 1

}

}

right\_eye\_center\_y\_testerror = as.numeric(right\_eye\_center\_y\_testerror\_MSE/count)

right\_eye\_center\_y\_testerror = sqrt(right\_eye\_center\_y\_testerror)

right\_eye\_center\_y\_testerror

#calculate total test error for right\_eye\_center

right\_eye\_center\_testerror = as.numeric(((right\_eye\_center\_x\_testerror^2)+(right\_eye\_center\_y\_testerror^2))/2)

right\_eye\_center\_testerror = sqrt(right\_eye\_center\_testerror)

right\_eye\_center\_testerror

rec\_x <- candidate\_pixel\_and\_score\_righteyecenter$x

rec\_y <- candidate\_pixel\_and\_score\_righteyecenter$y

rec\_trainingerror <- right\_eye\_center\_trainingerror

rec\_testerror <- right\_eye\_center\_testerror

data.frame(search\_size,rec\_x, rec\_y, rec\_trainingerror, rec\_testerror)

}

#print the training and test errors for right eye center to an output file

write.csv(output,"~/vyshya/IUPUI/mybooks/fall2015/intelligent\_systems/prject/data/outputErrors\_righteyecenter.csv")

################################Right Eye Center##################################################

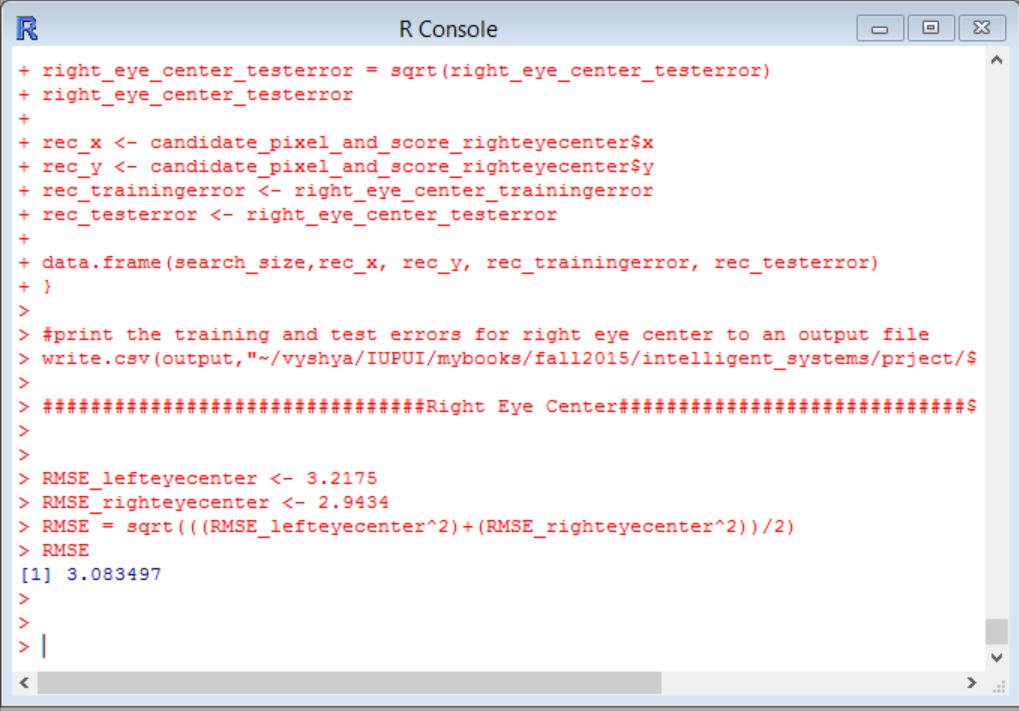
RMSE\_lefteyecenter <- 3.2175

RMSE\_righteyecenter <- 2.9434

RMSE = sqrt(((RMSE\_lefteyecenter^2)+(RMSE\_righteyecenter^2))/2)

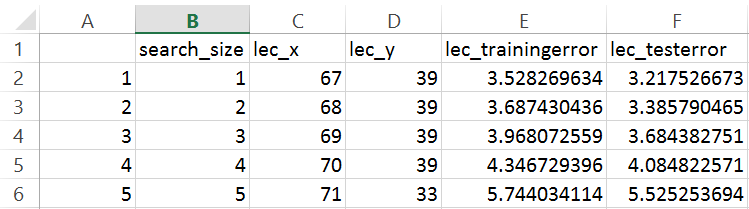
RMSE

**Output:**

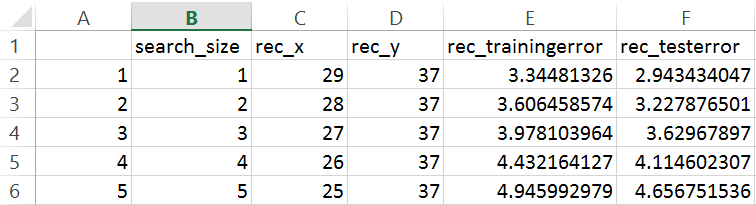


**Generated Output Files:**

**outputErrors\_lefteyecenter.csv:**

****

**outputErrors\_righteyecenter.csv:**

****