Assignment 2

Random Forests

Sharin KG

14111033

Question 1.

As discussed in class build a random forest classifier using bagging and random splitting using m random features to decide the split at each node. Find the number of trees at which the error levels off by using a binary search between 2 trees and 400-500 trees. Plot 5-fold cross-validated error rates against the number of trees in the forest and report the number at which error levels off.

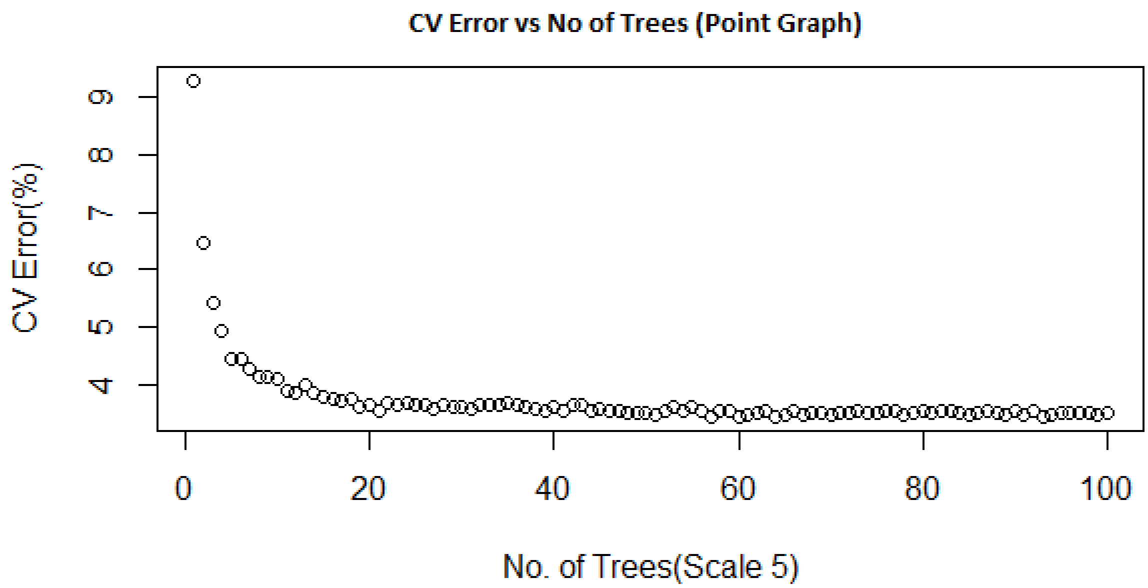
Solution

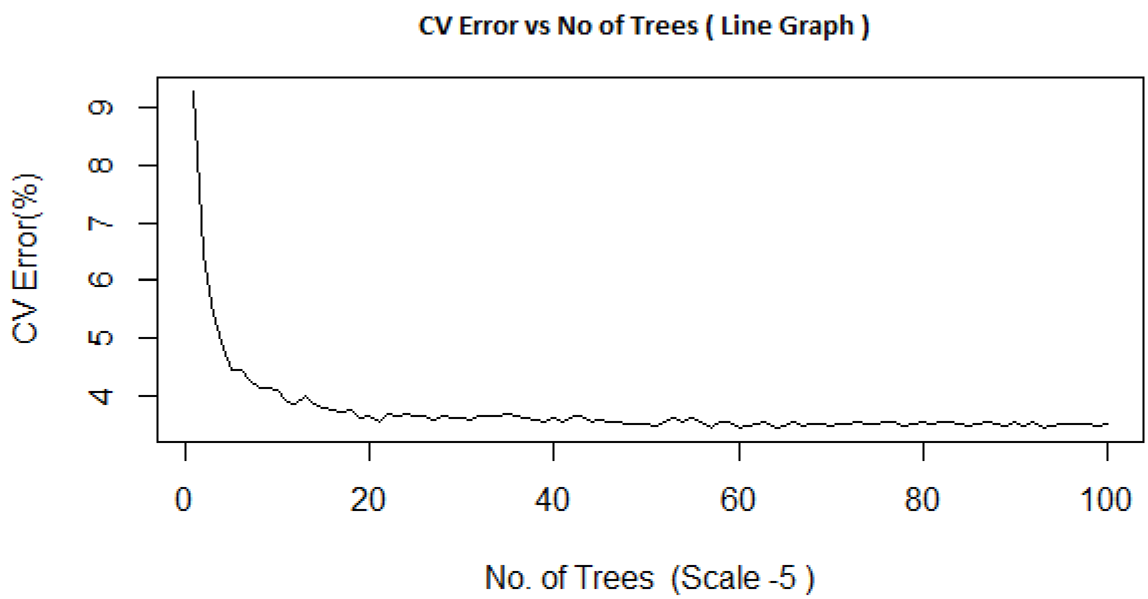
The number of trees at which the error levels off was found to be **93** by using Binary search between **2** and **500** trees. And the threshold for carrying out binary search was fixed to be **0.0015** based on the error rate at mid and last point.

```
#Function - Binary Search
binSearch <- function(h_error,low,high){
  mid <- floor((low+high)/2)
  m_error <- getXError(mid)
  int = abs(h_error - m_error)
  if((abs(low-mid)<=1)||(abs(mid-high)<=1)){
    return(high)
  }
  if(int<0.0015){
  high =mid
  return (binSearch(h_error,low,high))
  }
  else{
    low = mid
    return (binSearch(h_error,low,high))
  }
}

#Function - Validation Error
getXError <- function(noTree){
  set.seed(2)
  error<-numeric(0)
  for(i in 1:5)
  {
    x_test_s = folds[[i]]
    x_train_s =setdiff(c(1:20000), x_test_s)
    x_test = Data[x_test_s,]
    x_train = Data[x_train_s,]
    actual  = Data$lettr[x_test_s]
    x_fit <- randomForest(lettr ~ . ,data=x_train, ntree=noTree,replace = TRUE)
    predictions=predict(x_fit,newdata = x_test,type="class")
    error2=mean(actual!=predictions)
    error=rbind(error,error2)
  }
  return(mean(error))
}
```

Code Snippet 1. Function for Binary Search and Validation Error


CV Error vs No of Trees (Point Graph)
CV Error(%)
4
5
6
7
8
9
0
20
40
60
80
100
No. of Trees(Scale 5)


CV Error vs No of Trees (Line Graph)
CV Error(%)
4
5
6
7
8
9
0
20
40
60
80
100
No. of Trees (Scale -5)

Question 2.

Compute and report the out-of-bag error for the forest with the least error in part a).

Solution

The OOB for Number of Trees – 93 was found to be

```
set.seed(415)
rf=randomForest(lettr~.,data=Data,ntree = 93,replace = TRUE)
print(rf)
```

```
OOB estimate of  error rate: 3.63%
```

Code Snippet 2. Finding OOB Rate

Question 3.

Experiment with different values of m (say 1, 2, 4, 8) and report 5 fold cross-validated error rates in the form of a table fixing the number of trees at 1.25 times the number obtained in part a) (where the error levels off).

Solution

Fixing the Number of Trees to be – 93*1.25 = 116, Cross Validated errors were generated.

As the number of attributes was 17, the least error rate was expected be around 4 (~ root (17)) and this was found to be true.

m Value	5 Fold Cross Validated Error Value (%)
1	5.60
2	3.87
4	3.72
8	4.16

Table 1. m vs. CV error rate

```
m=1
total_error<-array(0,dim=4)
for(k in 1:4){
 set.seed(415)
 cerror<-numeric(0)
 print("Count")
 print(m)
 for(i in 1:5) {
   cx_test_s = folds[[i]]
   cx_train_s =setdiff(c(1:20000), cx_test_s)
   cx_test = Data[cx_test_s,]
   cx_train = Data[cx_train_s,]
   cactual  = Data$lettr[cx_test_s]
   cx_fit <- randomForest(lettr ~ . ,data=cx_train, ntree=116,replace = TRUE,mtry = m)
   prediction=predict(cx_fit,newdata = cx_test,type="class")
   cerror=rbind(cerror,mean(cactual!=prediction))
 }
 print(cerror)
 total_error[k]=mean(cerror)
 m=m*2
}
print(total_error*100)
```

Code Snippet 3. Generating CV Error rates for m=1,2,3,4

Question 4.

Study the effect of the size of the randomly sampled data set from L while constructing a tree. Start by sampling 10% of the points from L for constructing a tree and go up to 80% in increments of 10% and the number of trees, say T, in the forest at which the error levels off in each case. Find the 5 fold cross-validated error for a forest with T trees and plot it against the size of the sampled data set expressed as a percentage of L. Comment on the results and say whether bagging is justified as a randomization method to select samples?

Solution

For sampling size ranging from 10% to 80% of the training set, the CV error rate vs. Size of sampled data plots were made, after finding the error off number of trees for each time.

- ***CV Error vs. No of Trees Plot***

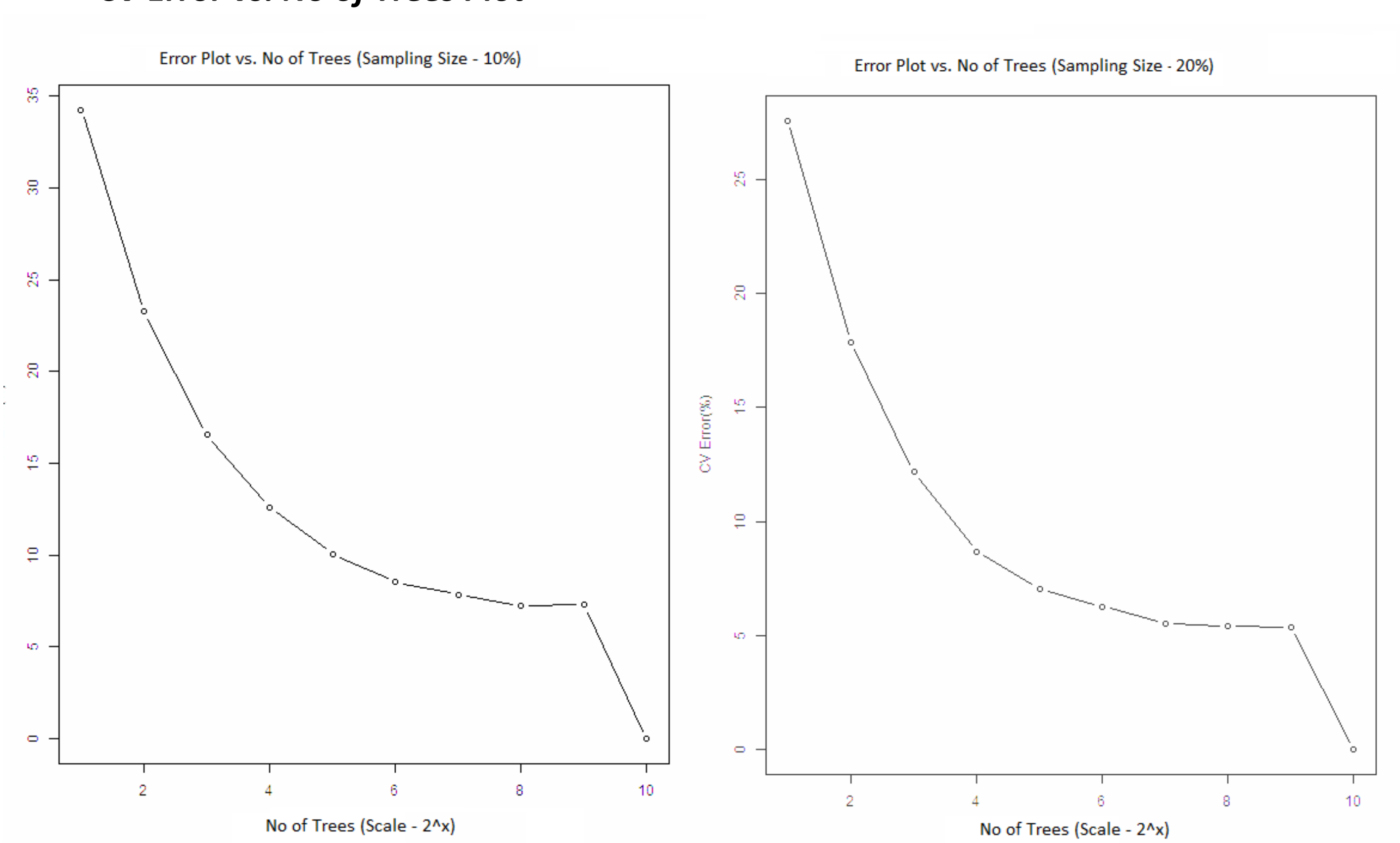


Figure 2: For Sampling Rates 10 and 20%

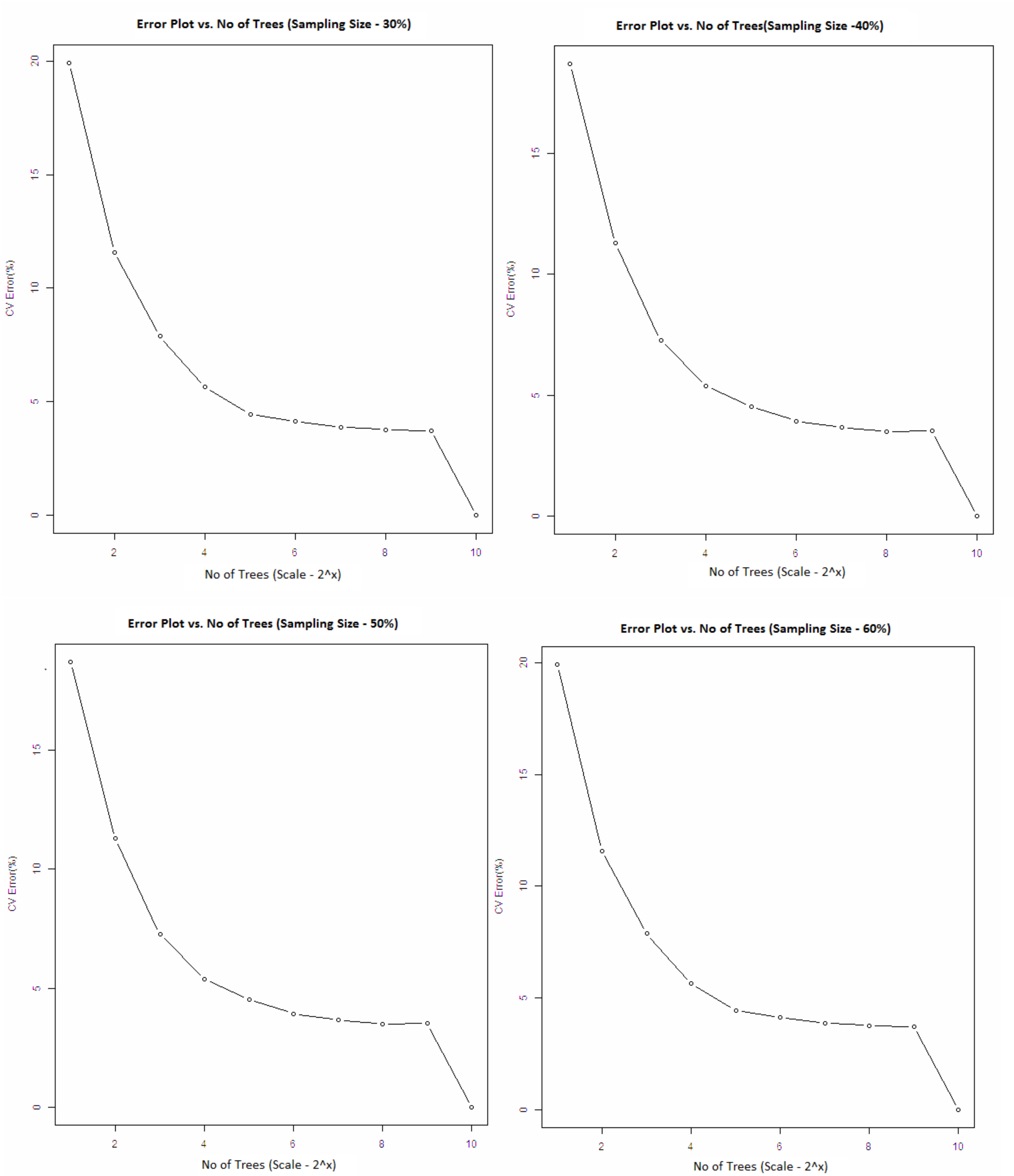


Figure 2: For Sampling Rates 30 -60%

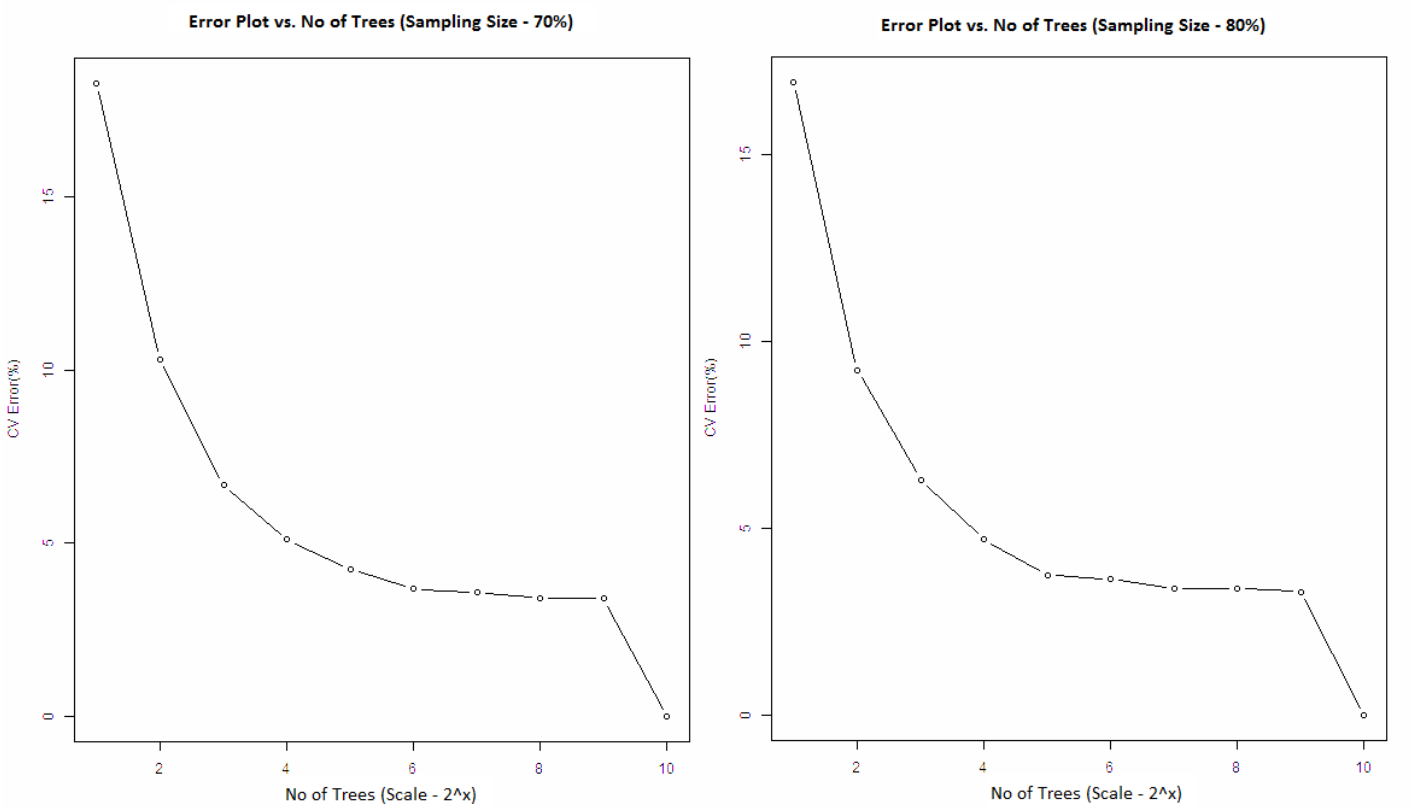
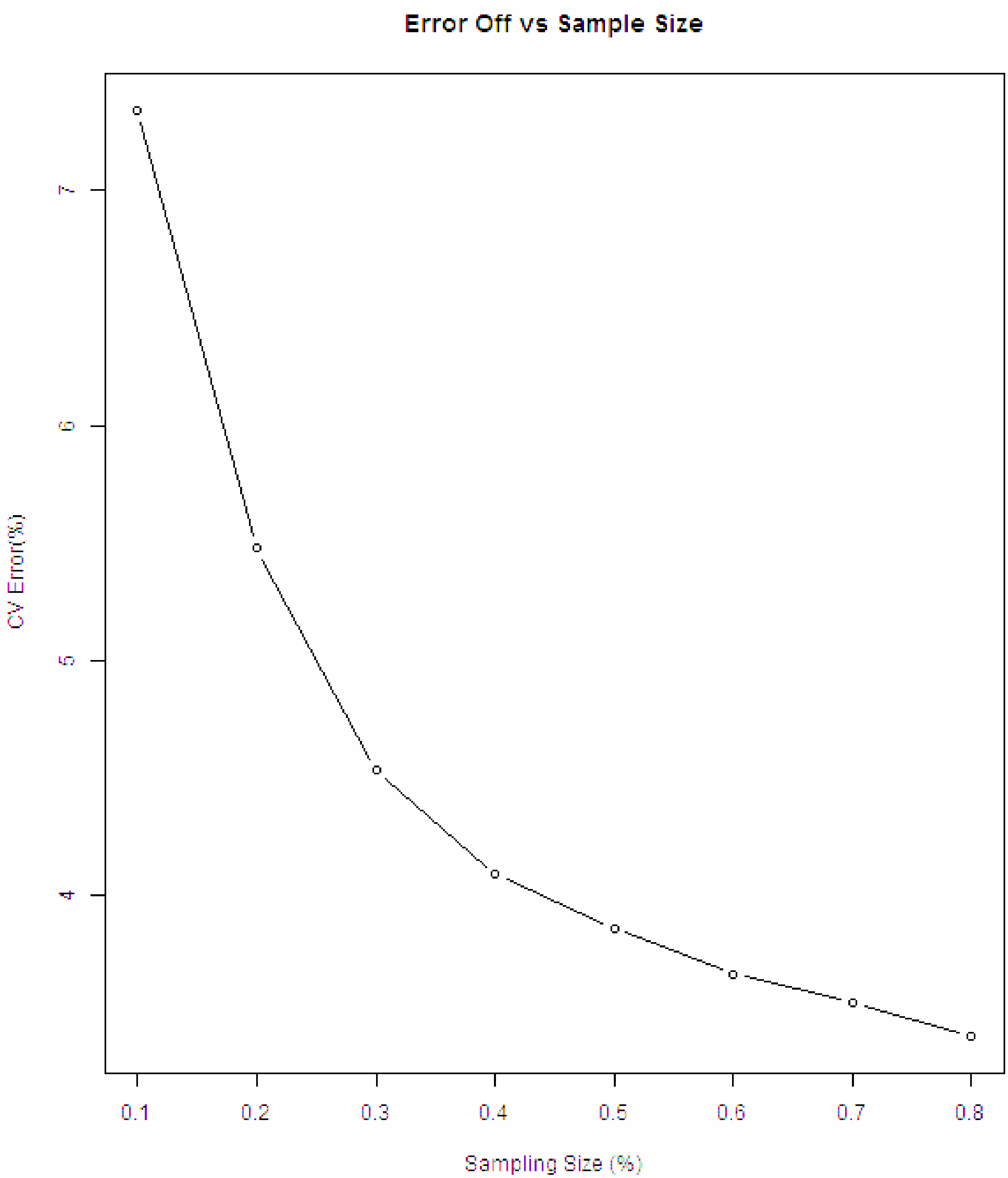


Figure 4: For Sampling Rates 70 and 80%

From the above and below graphs, we can clearly see the trend that is – As the size of the sampled data from the Learning Set increases, the error rate clearly decreases up to a certain point after which it becomes stable (around 60%), and since bagging also takes into account ~63% of randomized data, it is justified.

- *5 fold CV vs. Size of the Sampled Data from L*



```
#Function - Validation Error
getXError <- function(noTree,size){
  set.seed(415)
  error<-numeric(0)
  for(i in 1:5){
    x_test_s = folds[[i]]
    x_train_s =setdiff(c(1:20000), x_test_s)
    x_test = Data[x_test_s,]
    x_train = Data[x_train_s,]
    actual  = Data$lettr[x_test_s]
    x_fit <- randomForest(lettr ~ . ,data=x_train, ntree=noTree,sampsize =
nrow(x_train)*size,replace = FALSE)
    predictions=predict(x_fit,newdata = x_test,type="class")
    error=rbind(error,mean(actual!=predictions))
  }
  return(mean(error))
}
#Function - Binary Search
binSearch <- function(h_error,low,high,ssize){
  mid <- floor((low+high)/2)
  m_error <- getXError(mid,ssize)
  int = abs(h_error - m_error)
  if((abs(low-mid)<=1)||(abs(mid-high)<=1)){
    return(high)
  }
  if(int<0.0015){
    high =mid
    return (binSearch(h_error,low,high,ssize))
  }
  else{
    low = mid
    return (binSearch(h_error,low,high,ssize))
  }
}
arr_opt_pt<-array(0,dim=8)
arr_x<-array(0,dim=8)
for(k in 1:8){
  s_size = sample_size[k]
  tresh = getXError(500,s_size)
  arr_opt_pt[k]=binSearch(tresh,2,500,s_size)
  arr_x[k]<- getXError(arr_opt_pt[k],s_size)
  error_plot=array(0,dim=10)
  for(i in 1:9){
    if(i<9){
      error_plot[i]=getXError(2^i,s_size)
    }
    else{
      error_plot[i]=getXError(500,s_size)
    }
  }
  fileName = paste("Plot_", k, ".jpg")
  png(filename = fileName, width = 600, height = 700)
  plot(error_plot*100,type = "b",xlab = "No. of Trees  (Scale - 2^x)",ylab = "CV Error(%)", main
= "Error Plot vs No. of Trees")
  dev.off()
}
png(filename = "Part4.jpg", width = 600, height = 700)
plot(x=sample_size, y=arr_x*100,type = "b",xlab = "Sampling Size (%)",ylab = "CV Error(%)",
main="Error Off vs Sample Size")
dev.off()
```

Code Snippet for Part d of the Question