Pre-process & Split data based on EDA

```
> ###############################################################################
> #                                  EDA                                        #
> ###############################################################################
>
> datasetFile = "letter-recognition.data"
> dataset = read.csv(datasetFile, header=FALSE, sep=",")
> # Change the dependent variable column name to Letter
> colnames(dataset)[1] <- "Letter"
>
> # convert letters to number
> dataset$Letter = factor(dataset$Letter,
+                         levels = c("A",
 "B","C","D","E","F","G","H","I","J","K","L","M","N","O","P",
+                                    "Q","R","S","T","U","V","W","X","Y","Z"),
+                         labels = c(1, 2, 3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,
24,25,26))
>
> dataFeatureVariable = dataset[,-which(names(dataset)=='Letter')]
>
> # check for outliers
> outliers_row=c() # Loop over the feature columns
> for(i in colnames(dataFeatureVariable)) {
+   data_mean=mean(dataset[,i]) # Mean of the data in feature column i
+   data_sd=sd(dataset[,i]) # Standard deviation of the data in feature column i
+   low_cutoff=data_mean-3*data_sd # Lower cutoff value
+   upper_cutoff=data_mean+3*data_sd # Upper cutoff value
+   outliers_idx=which(dataset[,i]<low_cutoff | dataset[,i]>upper_cutoff)
+   outliers_row=c(outliers_row,outliers_idx)
+ }
> outliers_row=unique(outliers_row) # Remove duplicated row indices
> print(paste("Number of Outilers =",length(outliers_row)))
[1] "Number of Outilers = 1736"
>
```

From above we observed that in the dataset there are **1736 outliers** so before applying the algorithm we will remove these outliers from our dataset and we will then divide this data set into training and testing data set which is as shown below.

```
> dataset_without_outliers <- dataset[-c(outliers_row),]
>
> set.seed(43)
> randomized__withoutOutliers=dataset_without_outliers[sample(1:nrow(dataset_without_outliers),nrow
(dataset_without_outliers)),]
> tridx_withoutOutliers=sample(1:nrow(dataset_without_outliers),0.7*nrow(dataset_without_outliers),
replace=F)
> trainingDataset_withoutOutliers = randomized__withoutOutliers[tridx_withoutOutliers,]
> testingDataset_withoutOutliers = randomized__withoutOutliers[-tridx_withoutOutliers,]
```

Mohit Shankar Pursnani **Letter Recognition Dataset** mp5578@nyu.edu

Assignment - 3

For this Assignment, we will run the following Ensemble Techniques on our data set:

- Cross Validation
- Bagging
- Random Forest
- Boosting GBM

Cross Validation

Let's first look at **Cross Validation**. Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model. We will run CV on LDA, QDA, SVM, Tree and KNN. I have created a method which takes in type of the model, dataset numberofFolds, dataType as input as shown below.

```
> ##############################################################################
> #                        LDA, QDA, SVM, Tree CV                              #
> ##############################################################################
> trdf <- trainingDataset_withoutOutliers
> tstdf <- testingDataset_withoutOutliers
>
> crossValidationModels=function(df,type,numberFolds,dataType){
+   cl<-makePSOCKcluster(5)
+   registerDoParallel(cl)
+   start.time<-proc.time()
+   N<-nrow(df)
+   NF=numberFolds
+   folds<-split(1:N,cut(1:N, quantile(1:N, probs = seq(0, 1, by =1/NF))))
+   ridx<-sample(1:nrow(df),nrow(df),replace=FALSE)
+   # For lda
+   if(type=="lda"){
+     cv_df<-do.call('rbind',lapply(folds,FUN=function(idx,data=trdf[ridx,]) {
+       m <- lda(Letter~., data = df[-idx,])
+       p <- predict(m, df[idx,], type='response')
+       pred_tbl<-table(trdf[idx,c(which(names(trdf)=="Letter"))],p$class, dnn = c('Actual Group','Predicte
d Group'))
+       pred_cfm<-caret::confusionMatrix(pred_tbl)
+       list(cfm=pred_cfm) # store the fold, model,cfm
+     }))
+   }
+   # For qda
+   else if(type=="qda"){
+     cv_df<-do.call('rbind',lapply(folds,FUN=function(idx,data=trdf[ridx,]) {
+       m <- qda(Letter~., data = df[-idx,])
+       p <- predict(m, df[idx,], type='response')
+       pred_tbl<-table(trdf[idx,c(which(names(trdf)=="Letter"))],p$class, dnn = c('Actual Group','Predicte
d Group'))
+       pred_cfm<-caret::confusionMatrix(pred_tbl)
+       list(cfm=pred_cfm) # store the fold, model,cfm
+     }))
+   }
+   # For SVM
+   else if(type=="svm"){
+     cv_df<-do.call('rbind',lapply(folds,FUN=function(idx,data=trdf[ridx,]) {
+       m <- svm(Letter~., data = df[-idx,])
+       p <- predict(m, df[idx,],type="class")
+       pred_tbl<-table(trdf[idx,c(which(names(trdf)=="Letter"))],p, dnn = c('Actual Group','Predicted Grou
p'))
+       pred_cfm<-caret::confusionMatrix(pred_tbl)
+       list(cfm=pred_cfm) # store the fold, model,cfm
+     }))
+   }
+   # For Tree
+   else if(type=="tree"){
+     cv_df<-do.call('rbind',lapply(folds,FUN=function(idx,data=df[ridx,]) {
+       m <- rpart(Letter~., data = df[-idx,])
+       temp_df <- df[, -which(names(trainingDataset_withoutOutliers)=="Letter")]
+       p <- predict(m, temp_df[idx,],type="class")
+       pred_tbl<-table(df[idx,c(which(names(df)=="Letter"))],p, dnn = c('Actual Group','Predicted Group'))
+       pred_cfm<-caret::confusionMatrix(pred_tbl)
+       list(cfm=pred_cfm) # store the fold, model,cfm
+     }))
+   }
+   # For KNN
+   else if(type=="knn"){
+     cv_df<-do.call('rbind',lapply(folds,FUN=function(idx,data=df[ridx,]) {
+       trdf_knn=df[-idx, -which(names(df)=="Letter")]
+       tstdf_knn=df[idx, -which(names(df)=="Letter")]
+       trclass_knn=factor(df[-idx, which(names(df)=="Letter")])
+       tstclass_knn=factor(df[idx, which(names(df)=="Letter")])
+       knn_pred=knn(trdf_knn,tstdf_knn,trclass_knn, k = 1)
+       knn_cfm_tst=confusionMatrix(table(tstclass_knn,knn_pred))
+       list(fold=idx,knn_pred=knn_pred,cfm=knn_cfm_tst)
+     }))
+   }
+   else {
+     print("type should be 'lda' 'qda' 'knn' 'tree' 'svm'")
+     return()
+   }
+   cv_df<-as.data.frame(cv_df)
+   trcv.perf<-as.data.frame(do.call('rbind',lapply(cv_df$cfm,FUN=function(cfm)c(cfm$overall))))
```

```
acc_varEstp <- trcv.perf$Accuracy
mean_varEstp = signif(mean(acc_varEstp),4)
var_varEstp = signif(var(acc_varEstp),4)
varEstp = data.frame(mean_varEstp,var_varEstp)
names(varEstp) = c("Mean of Accuracies","Variance of Accuracies")
varianceEstimation <- t(varEstp)
print("varianceEstimation")
print(varianceEstimation)
print("Mean")
if(dataType == "train")
{
  (cv.tr.perf<-apply(trcv.perf[trcv.perf$AccuracyPValue<0.01,-c(6:7)],2,mean))
  print(cv.tr.perf)
  print("Standard Deviation")
  (cv.tr.perf.sd<-apply(trcv.perf[trcv.perf$AccuracyPValue<0.01,-c(6:7)],2,sd))
  print(cv.tr.perf.sd)
  print("Variance")
  (cv.tr.perf.var<-apply(trcv.perf[trcv.perf$AccuracyPValue<0.01,-c(6:7)],2,var))
  print(cv.tr.perf.var)
}
else
{
  (cv.tr.perf<-apply(trcv.perf[trcv.perf$AccuracyPValue>0.01,-c(6:7)],2,mean))
  print(cv.tr.perf)
  print("Standard Deviation")
  (cv.tr.perf.sd<-apply(trcv.perf[trcv.perf$AccuracyPValue>0.01,-c(6:7)],2,sd))
  print(cv.tr.perf.sd)
  print("Variance")
  (cv.tr.perf.var<-apply(trcv.perf[trcv.perf$AccuracyPValue>0.01,-c(6:7)],2,var))
  print(cv.tr.perf.var)
}
confusion_matrix <- cv_df$cfm
cfm_len <- length(cv_df$cfm)
for(c in confusion_matrix) {
  confusionMatrixByClass <- c$byClass
  confusion_matrix_sd <- c$byClass
  break
}
y <- 0
z <- c()
 for(column in colnames(confusionMatrixByClass)) {
   for(row in rownames(confusionMatrixByClass)) {
     for (x in confusion_matrix) {
       my_mat <- as.matrix(x$byClass)
       y <- y + my_mat[row,column]
       z <- c(z,my_mat[row,column])
     }
     confusionMatrixByClass[row,column] <- y/cfm_len
     confusion_matrix_sd[row, column] <- sd(z)
     y <- 0
     z <- c()
   }
 }
 print(confusionMatrixByClass)
 stopCluster(cl)
}
```

We know that **Cross Validation does not help to improve the accuracy of the model, it generally results in a less biased or less optimistic estimate of the model skill than other methods**. Let's see the behavior of Cross Validation for training dataset with the LDA model. Variance of accuracy for training set is **0.0002802**. This variance shows that Cross validation has little effect on model's accuracy with different data set as it is very close to zero.

```
> crossValidationModels(trdf, "lda", 20, "train")
[1] "varianceEstimation"
                            [,1]
Mean of Accuracies     0.7177000
Variance of Accuracies 0.0002802
[1] "Mean"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
   0.71767373    0.70600327    0.68107161    0.75225099    0.06180115
[1] "Standard Deviation"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
  0.016740140   0.017374653   0.017279038   0.016044402   0.005802799
[1] "Variance"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
 2.802323e-04  3.018786e-04  2.985652e-04  2.574228e-04  3.367247e-05
```

```
          Sensitivity Specificity Pos Pred Value Neg Pred Value Precision    Recall        F1
Class: 1    0.9054342   0.9941399      0.8616443      0.9961694 0.8616443 0.9054342 0.8811254
Class: 2    0.5643881   0.9889728      0.7557569      0.9744293 0.7557569 0.5643881 0.6426662
Class: 3    0.7025087   0.9902882      0.7612232      0.9876441 0.7612232 0.7025087 0.7258640
Class: 4    0.7262127   0.9890270      0.7535684      0.9872555 0.7535684 0.7262127 0.7364681
Class: 5    0.6628811   0.9818564      0.5391431      0.9893612 0.5391431 0.6628811 0.5842178
Class: 6    0.6802943   0.9867322      0.7080550      0.9854647 0.7080550 0.6802943 0.6885315
Class: 7    0.6106667   0.9805566      0.5291314      0.9859118 0.5291314 0.6106667 0.5619829
Class: 8    0.3508417   0.9819970      0.4376759      0.9736692 0.4376759 0.3508417 0.3855088
Class: 9    0.8734874   0.9917922      0.8132113      0.9949446 0.8132113 0.8734874 0.8371129
Class: 10   0.8409884   0.9884647      0.6978049      0.9949672 0.6978049 0.8409884 0.7586317
Class: 11   0.6945769   0.9864211      0.6714016      0.9875482 0.6714016 0.6945769 0.6760856
Class: 12   0.8847114   0.9907593      0.7242675      0.9967678 0.7242675 0.8847114 0.7912538
Class: 13   0.8869391   0.9962851      0.8882148      0.9960317 0.8882148 0.8869391 0.8857588
Class: 14   0.7373768   0.9923706      0.7852207      0.9908570 0.7852207 0.7373768 0.7561628
Class: 15   0.6097429   0.9900228      0.7683167      0.9791294 0.7683167 0.6097429 0.6753935
Class: 16   0.8473326   0.9890422      0.7440702      0.9941186 0.7440702 0.8473326 0.7892238
Class: 17   0.6677153   0.9850474      0.6314088      0.9869002 0.6314088 0.6677153 0.6449883
Class: 18   0.6481701   0.9896027      0.7422016      0.9837948 0.7422016 0.6481701 0.6903541
Class: 19   0.4818346   0.9809217      0.5609153      0.9735574 0.5609153 0.4818346 0.5128959
Class: 20   0.8356712   0.9884239      0.7192882      0.9941367 0.7192882 0.8356712 0.7706152
Class: 21   0.8138948   0.9923048      0.7983835      0.9934334 0.7983835 0.8138948 0.8008227
Class: 22   0.8223797   0.9939517      0.8619489      0.9921775 0.8619489 0.8223797 0.8381155
Class: 23   0.8332294   0.9960213      0.8872062      0.9936796 0.8872062 0.8332294 0.8579907
Class: 24   0.6895378   0.9890107      0.7538536      0.9851464 0.7538536 0.6895378 0.7159269
Class: 25   0.8681731   0.9819045      0.5750249      0.9962466 0.5750249 0.8681731 0.6877404
Class: 26   0.7708271   0.9904970      0.7063037      0.9929698 0.7063037 0.7708271 0.7303118
          Prevalence Detection Rate Detection Prevalence Balanced Accuracy
Class: 1  0.03841134     0.03473469           0.04036691         0.9497870
Class: 2  0.05656005     0.03207453           0.04247897         0.7766805
Class: 3  0.04091354     0.02902264           0.03833187         0.8463984
Class: 4  0.04451205     0.03230768           0.04279098         0.8576199
Class: 5  0.02996173     0.01971391           0.03731392         0.8223687
Class: 6  0.04326095     0.02933563           0.04200924         0.8335133
Class: 7  0.03473249     0.02119938           0.03997433         0.7956116
Class: 8  0.03942696     0.01392471           0.03121283         0.6664194
Class: 9  0.03856759     0.03371785           0.04162021         0.9326398
Class: 10 0.03035297     0.02550262           0.03668843         0.9147266
Class: 11 0.03825423     0.02628521           0.03934871         0.8404990
Class: 12 0.02667645     0.02354741           0.03254365         0.9377354
Class: 13 0.03395088     0.03011737           0.03371601         0.9416121
Class: 14 0.03442024     0.02558037           0.03293464         0.8648737
Class: 15 0.05163097     0.03160456           0.04107101         0.7998829
Class: 16 0.03739302     0.03176056           0.04232113         0.9181874
Class: 17 0.03762789     0.02503350           0.03942745         0.8263814
Class: 18 0.04482529     0.02925861           0.03919356         0.8188864
Class: 19 0.04858042     0.02323393           0.04138302         0.7313782
Class: 20 0.03434309     0.02871014           0.03989669         0.9120476
Class: 21 0.03449922     0.02816266           0.03559358         0.9030998
Class: 22 0.04185214     0.03434199           0.04013119         0.9081657
Class: 23 0.03723665     0.03113507           0.03496833         0.9146254
Class: 24 0.04529306     0.03105609           0.04153829         0.8392743
Class: 25 0.02706719     0.02346843           0.04107028         0.9250388
```

The accuracy of the model for testing dataset has reduced drastically. The test accuracy with CV is 0.035 and variance is in the range of e-05, which shows that the we need to improve the model or the model chosen is not good enough. The accuracy drop with the test dataset shows that **the model is over-fitting**. Judging accuracy based on just training set will be wrong. This drop can also be because the dependent variables depend on one or two features as we saw in Assignment 2 when we ran LDA using Predictor Interaction.

```
> crossValidationModels(tstdf, "lda", 20, "test")
[1] "varianceEstimation"
                          [,1]
Mean of Accuracies     3.504e-02
Variance of Accuracies 6.775e-05
[1] "Mean"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
  0.035040507  -0.003951006   0.016792425   0.064100752   0.069171947
[1] "Standard Deviation"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
  0.008231066   0.008511699   0.005603749   0.010550614   0.009045409
[1] "Variance"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
 6.775045e-05  7.244903e-05  3.140200e-05  1.113155e-04  8.181942e-05
           Sensitivity Specificity Pos Pred Value Neg Pred Value   Precision      Recall  F1 Prevalence
Class: 1  0.037202748   0.9561441     0.034044567      0.9614356 0.034044567 0.037202748 NaN 0.03832785
Class: 2  0.048324587   0.9545007     0.051340902      0.9497627 0.051340902 0.048324587 NaN 0.05019385
Class: 3  0.025972222   0.9617733     0.031083916      0.9544718 0.031083916 0.025972222 NaN 0.04507834
Class: 4  0.024081197   0.9563271     0.027420635      0.9524968 0.027420635 0.024081197 NaN 0.04672068
Class: 5  0.017670455   0.9602637     0.011868687      0.9677392 0.011868687 0.017670455 NaN 0.03158004
Class: 6  0.023750367   0.9584918     0.024783550      0.9628644 0.024783550 0.023750367 NaN 0.03668418
Class: 7  0.039522561   0.9585273     0.041640720      0.9585280 0.041640720 0.039522561 NaN 0.04143004
Class: 8  0.041279762   0.9677821     0.044226190      0.9630394 0.044226190 0.041279762 NaN 0.03723229
Class: 9  0.043412698   0.9586155     0.043132215      0.9622550 0.043132215 0.043412698 NaN 0.03778174
Class: 10 0.056628788   0.9668622     0.044148441      0.9756276 0.044148441 0.056628788 NaN 0.02500668
Class: 11 0.017222222   0.9621515     0.021410534      0.9577651 0.021410534 0.017222222 NaN 0.04143205
Class: 12 0.024305556   0.9683864     0.020972222      0.9762310 0.020972222 0.024305556 NaN 0.02372931
Class: 13 0.044930556   0.9659356     0.036953782      0.9646739 0.036953782 0.044930556 NaN 0.03558595
Class: 14 0.029739011   0.9680556     0.027771465      0.9656904 0.027771465 0.029739011 NaN 0.03413010
Class: 15 0.038879400   0.9549000     0.041312690      0.9510720 0.041312690 0.038879400 NaN 0.04872931
Class: 16 0.071746032   0.9582902     0.051479950      0.9636033 0.051479950 0.071746032 NaN 0.03705048
Class: 17 0.027172619   0.9584397     0.021445360      0.9573474 0.021445360 0.027172619 NaN 0.04179835
Class: 18 0.045982302   0.9601302     0.045754731      0.9510245 0.045754731 0.045982302 NaN 0.04909561
Class: 19 0.053273810   0.9590534     0.046738678      0.9636151 0.046738678 0.053273810 NaN 0.03649902
Class: 20 0.036885476   0.9608691     0.030449594      0.9658051 0.030449594 0.036885476 NaN 0.03412877
Class: 21 0.045202020   0.9639444     0.047857143      0.9665008 0.047857143 0.045202020 NaN 0.03376514
Class: 22 0.034964402   0.9610006     0.034982517      0.9538854 0.034982517 0.034964402 NaN 0.04581361
Class: 23 0.029185259   0.9673274     0.041739927      0.9603869 0.041739927 0.029185259 NaN 0.03960656
Class: 24 0.031944444   0.9597038     0.041014264      0.9502119 0.041014264 0.031944444 NaN 0.04964038
Class: 25 0.022222222   0.9599231     0.019090909      0.9745093 0.019090909 0.022222222 NaN 0.02519117
Class: 26 0.009401709   0.9685937     0.008571429      0.9654965 0.008571429 0.009401709 NaN 0.03376848
```

```
         Detection Rate Detection Prevalence Balanced Accuracy
Class: 1     0.0014598540          0.04362116         0.4966734
Class: 2     0.0021904495          0.04544598         0.5014126
Class: 3     0.0012773723          0.03777773         0.4938728
Class: 4     0.0012773723          0.04288923         0.4902042
Class: 5     0.0005481137          0.03905845         0.4889671
Class: 6     0.0010948905          0.04106441         0.4911211
Class: 7     0.0016423358          0.04143405         0.4990250
Class: 8     0.0014598540          0.03248710         0.5045309
Class: 9     0.0016423358          0.04142937         0.5010141
Class: 10    0.0014598540          0.03376648         0.5117455
Class: 11    0.0007299270          0.03704914         0.4896868
Class: 12    0.0007299270          0.03157670         0.4963460
Class: 13    0.0014598540          0.03431259         0.5054331
Class: 14    0.0009124088          0.03175918         0.4988973
Class: 15    0.0020072993          0.04489920         0.4968897
Class: 16    0.0021904495          0.04234713         0.5150181
Class: 17    0.0009124088          0.04069744         0.4928061
Class: 18    0.0020079677          0.03996885         0.5030563
Class: 19    0.0016430042          0.04106107         0.5061636
Class: 20    0.0012773723          0.03905845         0.4988773
Class: 21    0.0014598540          0.03632657         0.5045732
Class: 22    0.0014605224          0.03869348         0.4979825
Class: 23    0.0012773723          0.03267025         0.4982563
Class: 24    0.0018248175          0.04015200         0.4958241
Class: 25    0.0007299270          0.03978971         0.4910727
Class: 26    0.0003649635          0.03066429         0.4889977
```

Cross Validation – QDA

Let's do Cross validation using QDA.

The accuracy of the model with CV is 0.8838 and variance is 0.000144. CV on QDA shows drop in accuracy for training data set.

```
> crossValidationModels(trdf, "qda", 20, "train")
[1] "varianceEstimation"
                           [,1]
Mean of Accuracies     0.883800
Variance of Accuracies 0.000144
[1] "Mean"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
   0.88382874    0.87898842    0.85645290    0.90759343    0.05671679
[1] "Standard Deviation"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
  0.012001476   0.012483779   0.013032053   0.010852835   0.005473403
[1] "Variance"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
 1.440354e-04  1.558447e-04  1.698344e-04  1.177840e-04  2.995814e-05
         Sensitivity Specificity Pos Pred Value Neg Pred Value Precision    Recall        F1
Class: 1   0.9569536   0.9974721      0.9420365      0.9982057 0.9420365 0.9569536 0.9484652
Class: 2   0.8623207   0.9949342      0.8839390      0.9940496 0.8839390 0.8623207 0.8704076
Class: 3   0.9109423   0.9960830      0.9079861      0.9964981 0.9079861 0.9109423 0.9077179
Class: 4   0.8403529   0.9954935      0.9001967      0.9922326 0.9001967 0.8403529 0.8669801
Class: 5   0.8627036   0.9948732      0.8683088      0.9943835 0.8683088 0.8627036 0.8628475
Class: 6   0.8362940   0.9949991      0.8878953      0.9924841 0.8878953 0.8362940 0.8586371
Class: 7   0.8612716   0.9941386      0.8583656      0.9942175 0.8583656 0.8612716 0.8573116
Class: 8   0.6719323   0.9887183      0.6476268      0.9897453 0.6476268 0.6719323 0.6501319
Class: 9   0.9507090   0.9939104      0.8616667      0.9981261 0.8616667 0.9507090 0.9008161
Class: 10  0.9403509   0.9970778      0.9229381      0.9977263 0.9229381 0.9403509 0.9303536
Class: 11  0.8436187   0.9934896      0.8414982      0.9934863 0.8414982 0.8436187 0.8386131
Class: 12  0.9250730   0.9959661      0.8764312      0.9974890 0.8764312 0.9250730 0.8978679
Class: 13  0.9455805   0.9979823      0.9383449      0.9980564 0.9383449 0.9455805 0.9407376
Class: 14  0.8863942   0.9963542      0.8939214      0.9961933 0.8939214 0.8863942 0.8882568
Class: 15  0.8182349   0.9954989      0.8943559      0.9914296 0.8943559 0.8182349 0.8520128
Class: 16  0.9493944   0.9955233      0.8955706      0.9978775 0.8955706 0.9493944 0.9206928
Class: 17  0.8908550   0.9941438      0.8569190      0.9955162 0.8569190 0.8908550 0.8718357
Class: 18  0.8316153   0.9960810      0.9045269      0.9924217 0.9045269 0.8316153 0.8642274
Class: 19  0.8461551   0.9948489      0.8840596      0.9930681 0.8840596 0.8461551 0.8615770
Class: 20  0.9172325   0.9944771      0.8662527      0.9966630 0.8662527 0.9172325 0.8889300
Class: 21  0.9074555   0.9980509      0.9483318      0.9963493 0.9483318 0.9074555 0.9252998
Class: 22  0.8827460   0.9964008      0.9194868      0.9948685 0.9194868 0.8827460 0.8986816
Class: 23  0.9224408   0.9978962      0.9376269      0.9970749 0.9376269 0.9224408 0.9290517
Class: 24  0.9333368   0.9940585      0.8654832      0.9973064 0.8654832 0.9333368 0.8963718
Class: 25  0.9184708   0.9931674      0.8434060      0.9967337 0.8434060 0.9184708 0.8775881
Class: 26  0.9083098   0.9975759      0.9262569      0.9969284 0.9262569 0.9083098 0.9156772
```

	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Class: 1	0.03966256	0.03794185	0.04036691	0.9772128
Class: 2	0.04333969	0.03762874	0.04247897	0.9286274
Class: 3	0.03794124	0.03457685	0.03833187	0.9535127
Class: 4	0.04592026	0.03848824	0.04279098	0.9179232
Class: 5	0.03778365	0.03238556	0.03731392	0.9287884
Class: 6	0.04443454	0.03723702	0.04200924	0.9156465
Class: 7	0.03989522	0.03434150	0.03997433	0.9277051
Class: 8	0.03019684	0.02026139	0.03121283	0.8303253
Class: 9	0.03755111	0.03575203	0.04162021	0.9723097
Class: 10	0.03606282	0.03387226	0.03668843	0.9687143
Class: 11	0.03934884	0.03309040	0.03934871	0.9185542
Class: 12	0.03105719	0.02863214	0.03254365	0.9605195
Class: 13	0.03363801	0.03176044	0.03371601	0.9717814
Class: 14	0.03309064	0.02941400	0.03293464	0.9413742
Class: 15	0.04498252	0.03676814	0.04107101	0.9068669
Class: 16	0.04005257	0.03801863	0.04232113	0.9724588
Class: 17	0.03809737	0.03379463	0.03942745	0.9424994
Class: 18	0.04271335	0.03543843	0.03919356	0.9138481
Class: 19	0.04310470	0.03645467	0.04138302	0.9205020
Class: 20	0.03778487	0.03457759	0.03989669	0.9558548
Class: 21	0.03723579	0.03371613	0.03559358	0.9527532
Class: 22	0.04161703	0.03668880	0.04013119	0.9395734
Class: 23	0.03575105	0.03293452	0.03496833	0.9601685
Class: 24	0.03840901	0.03582771	0.04153829	0.9636976
Class: 25	0.03762862	0.03449910	0.04107028	0.9558191
Class: 26	0.03270051	0.02972797	0.03207478	0.9529429

Let's see next the behavior of testing dataset and see if CV on QDA overfits the model.

The cross-validation accuracy on test dataset is 0.03778 using QDA, the accuracy drop is significant for testing dataset. This shows that we need to improve the model or the model chosen is not good enough. The model is overfitting this can be because of predictor interaction. Hence CV using QDA is also not well suited for this data set.

```
> crossValidationModels(tstdf, "qda", 20, "test")
[1] "varianceEstimation"
                          [,1]
Mean of Accuracies     0.0377800
Variance of Accuracies 0.0001518
[1] "Mean"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
  0.037780407  -0.001048049   0.018929806   0.067382165   0.067712093
[1] "Standard Deviation"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
  0.012320471   0.012944085   0.008225314   0.015944310   0.006841210
[1] "Variance"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
 1.517940e-04  1.675493e-04  6.765578e-05  2.542210e-04  4.680215e-05
```

	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1	Prevalence
Class: 1	0.04117586	0.9562673	0.03821123	0.9601004	0.03821123	0.04117586	NaN	0.03978837
Class: 2	0.04067335	0.9542903	0.03229328	0.9617751	0.03229328	0.04067335	NaN	0.03796690
Class: 3	0.04896355	0.9629298	0.05881896	0.9603645	0.05881896	0.04896355	NaN	0.04033248
Class: 4	0.02654211	0.9565482	0.03297619	0.9530756	0.03297619	0.02654211	NaN	0.04635504
Class: 5	0.02862034	0.9606412	0.02928807	0.9589992	0.02928807	0.02862034	NaN	0.04070078
Class: 6	0.02802510	0.9586148	0.03033911	0.9613429	0.03033911	0.02802510	NaN	0.03832852
Class: 7	0.06335859	0.9592695	0.05439637	0.9540036	0.05439637	0.06335859	NaN	0.04672201
Class: 8	0.04843434	0.9681100	0.04644841	0.9682956	0.04644841	0.04843434	NaN	0.03230395
Class: 9	0.04952076	0.9588332	0.04590999	0.9644858	0.04590999	0.04952076	NaN	0.03577311
Class: 10	0.05119228	0.9666509	0.04914844	0.9692079	0.04914844	0.05119228	NaN	0.03121240
Class: 11	0.01672619	0.9623378	0.02109002	0.9628746	0.02109002	0.01672619	NaN	0.03650437
Class: 12	0.02342949	0.9681457	0.02097222	0.9683282	0.02097222	0.02342949	NaN	0.03139822
Class: 13	0.03973050	0.9658669	0.03695378	0.9635074	0.03695378	0.03973050	NaN	0.03668084
Class: 14	0.03702922	0.9682645	0.03089646	0.9671796	0.03089646	0.03702922	NaN	0.03285407
Class: 15	0.04073649	0.9549723	0.03082739	0.9597366	0.03082739	0.04073649	NaN	0.04015066
Class: 16	0.04554258	0.9575705	0.03969262	0.9595826	0.03969262	0.04554258	NaN	0.04033649
Class: 17	0.02301282	0.9582671	0.01831224	0.9564457	0.01831224	0.02301282	NaN	0.04252627
Class: 18	0.03475936	0.9594914	0.03107830	0.9502114	0.03107830	0.03475936	NaN	0.04927542
Class: 19	0.03271520	0.9587070	0.02483211	0.9638158	0.02483211	0.03271520	NaN	0.03595559
Class: 20	0.02983211	0.9607013	0.02694842	0.9616256	0.02694842	0.02983211	NaN	0.03814337
Class: 21	0.04789412	0.9641764	0.05202381	0.9674613	0.05202381	0.04789412	NaN	0.03303388
Class: 22	0.04789377	0.9611482	0.03615822	0.9580564	0.03615822	0.04789377	NaN	0.04179835
Class: 23	0.03807595	0.9675874	0.04798993	0.9632025	0.04798993	0.03807595	NaN	0.03705115
Class: 24	0.03789918	0.9598987	0.03869103	0.9591720	0.03869103	0.03789918	NaN	0.04088327
Class: 25	0.04781677	0.9608283	0.05579115	0.9604513	0.05579115	0.04781677	NaN	0.04015534
Class: 26	0.01666667	0.9687918	0.01690476	0.9656935	0.01690476	0.01666667	NaN	0.03376915

```
          Detection Rate Detection Prevalence Balanced Accuracy
Class: 1   0.0016430042          0.04362116         0.4987216
Class: 2   0.0014598540          0.04544598         0.4974818
Class: 3   0.0021897810          0.03777773         0.5059467
Class: 4   0.0014598540          0.04288923         0.4915451
Class: 5   0.0012787091          0.03905845         0.4946307
Class: 6   0.0012780407          0.04106441         0.4933200
Class: 7   0.0025554130          0.04143405         0.5113140
Class: 8   0.0016423358          0.03248710         0.5082722
Class: 9   0.0018248175          0.04142937         0.5041770
Class: 10  0.0014598540          0.03376648         0.5089216
Class: 11  0.0007299270          0.03704914         0.4895320
Class: 12  0.0007299270          0.03157670         0.4957876
Class: 13  0.0014598540          0.03431259         0.5027987
Class: 14  0.0010955589          0.03175918         0.5026469
Class: 15  0.0016423358          0.04489920         0.4978544
Class: 16  0.0016430042          0.04234713         0.5015565
Class: 17  0.0007299270          0.04069744         0.4906399
Class: 18  0.0014605224          0.03996885         0.4971254
Class: 19  0.0012773723          0.04106107         0.4957111
Class: 20  0.0012773723          0.03905845         0.4952667
Class: 21  0.0016423358          0.03632657         0.5060353
Class: 22  0.0014598540          0.03869348         0.5045210
Class: 23  0.0014598540          0.03267025         0.5028317
Class: 24  0.0016430042          0.04015200         0.4988989
Class: 25  0.0021904495          0.03978971         0.5043225
Class: 26  0.0005474453          0.03066429         0.4927292
```

Cross Validation – Tree

Let's do Cross validation using Tree.

The accuracy of the model with CV is 0.4529 and variance is 0.0004239. CV on Tree show slight drop in accuracy for training data set. Without CV the model accuracy was 0.4567 and with CV the model's accuracy is 0.4529. As stated in Assignment 2, the tree model performs worst as with more class labels the prediction becomes more complex. Also, decision tree while building model used only some of the feature variables as Decision Tree is immune to multicollinearity, while building model it considers only few of the feature variables.

```
> crossValidationModels(trdf, "tree", 20, "train")
[1] "varianceEstimation"
                            [,1]
Mean of Accuracies      0.4529000
Variance of Accuracies 0.0004239
[1] "Mean"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
    0.4528664     0.4300016     0.4138203     0.4923498     0.1297796
[1] "Standard Deviation"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
   0.02058993    0.02104352    0.02031538    0.02067344    0.01608882
[1] "Variance"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
 0.0004239454  0.0004428299  0.0004127147  0.0004273911  0.0002588500
          Sensitivity Specificity Pos Pred Value Neg Pred Value  Precision    Recall        F1
Class: 1   0.9828205   0.9907877      0.7787347      0.9995138 0.77873474 0.9828205 0.8671292
Class: 2   0.3883298   0.9863605      0.7037192      0.9487877 0.70371922 0.3883298 0.4924835
Class: 3   0.7144388   0.9813670      0.5341451      0.9905481 0.53414513 0.7144388 0.5992227
Class: 4   0.2415395   0.9843426      0.6714584      0.9054924 0.67145836 0.2415395 0.3522996
Class: 5          NA   0.9711860             NA             NA 0.24872322        NA        NA
Class: 6          NA   0.9579908             NA             NA 0.00000000        NA        NA
Class: 7   0.3083434   0.9831926      0.6081651      0.9403227 0.60816509 0.3083434 0.4052989
Class: 8          NA   0.9687872             NA             NA 0.00000000        NA        NA
Class: 9   0.8722805   0.9900931      0.7751380      0.9945173 0.77513800 0.8722805 0.8148923
Class: 10  0.8787585   0.9866522      0.6474010      0.9965075 0.64740099 0.8787585 0.7413759
Class: 11         NA   0.9620187             NA             NA 0.04354167        NA        NA
Class: 12  0.5818103   0.9882960      0.6523087      0.9839035 0.65230870 0.5818103 0.6104903
Class: 13  0.9972222   0.9918217      0.7586373      0.9999192 0.75863734 0.9972222 0.8577033
Class: 14  0.6861293   0.9862217      0.5939524      0.9911853 0.59395239 0.6861293 0.6325259
Class: 15         NA   0.9746756             NA             NA 0.42439076        NA        NA
Class: 16  0.4305645   0.9798333      0.5444056      0.9680540 0.54440563 0.4305645 0.4780423
Class: 17         NA   0.9677993             NA             NA 0.17603577        NA        NA
Class: 18         NA   0.9608064             NA             NA 0.00000000        NA        NA
Class: 19  0.5356450   0.9678168      0.2425463      0.9893828 0.24254625 0.5356450 0.3277291
Class: 20  0.4780143   0.9860830      0.6686063      0.9692126 0.66860633 0.4780143 0.5545574
Class: 21  0.3677590   0.9848314      0.6009861      0.9622152 0.60098613 0.3677590 0.4491593
Class: 22  0.5091196   0.9829321      0.5997013      0.9760461 0.59970126 0.5091196 0.5457644
Class: 23  0.8306644   0.9920679      0.7791417      0.9941609 0.77914173 0.8306644 0.8015215
Class: 24  0.2517938   0.9813222      0.5936992      0.9172882 0.59369924 0.2517938 0.3415508
Class: 25         NA   0.9589297             NA             NA 0.00000000        NA        NA
Class: 26         NA   0.9709022             NA             NA 0.12493687        NA        NA
```

	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Class: 1	0.031918036	0.031448797	0.04036691	0.9868041
Class: 2	0.078936082	0.029883729	0.04247897	0.6873452
Class: 3	0.029334898	0.020260906	0.03833187	0.8479029
Class: 4	0.119372799	0.028944029	0.04279098	0.6129410
Class: 5	0.028629695	0.009229998	0.03731392	NA
Class: 6	0.000000000	0.000000000	0.04200924	NA
Class: 7	0.081825362	0.024562916	0.03997433	0.6457680
Class: 8	0.000000000	0.000000000	0.03121283	NA
Class: 9	0.037316975	0.032075020	0.04162021	0.9311868
Class: 10	0.027066828	0.023702807	0.03668843	0.9327053
Class: 11	0.008528951	0.001564945	0.03934871	NA
Class: 12	0.036846880	0.021278487	0.03254365	0.7850532
Class: 13	0.025815116	0.025736869	0.03371601	0.9945220
Class: 14	0.028083187	0.019556803	0.03293464	0.8361755
Class: 15	0.064462784	0.016898230	0.04107101	NA
Class: 16	0.053820667	0.023233324	0.04232113	0.7051989
Class: 17	0.023233690	0.007667009	0.03942745	NA
Class: 18	0.000000000	0.000000000	0.03919356	NA
Class: 19	0.020025919	0.009856710	0.04138302	0.7517309
Class: 20	0.056326536	0.026754695	0.03989669	0.7320486
Class: 21	0.057732908	0.021278487	0.03559358	0.6762952
Class: 22	0.046859351	0.023859057	0.04013119	0.7460259
Class: 23	0.032934884	0.027301814	0.03496833	0.9113661
Class: 24	0.104042694	0.024720144	0.04153829	0.6165580
Class: 25	0.000000000	0.000000000	0.04107028	NA
Class: 26	0.006885759	0.003051643	0.03207478	NA

Let's see next the behavior of CV on testing dataset.
The accuracy of CV on testing data set is slightly more than the accuracy of the model without CV. The accuracy with CV on testing data set is 0.4756 & without CV on testing data set the accuracy is 0.4567. But overall, the tree model is not able to capture the relationship accurately between the input and output accurately, generating a high error rate on both the training set and the testing set.

```
> crossValidationModels(tstdf, "tree", 20, "train")
[1] "varianceEstimation"
                          [,1]
Mean of Accuracies     0.475600
Variance of Accuracies 0.001795
[1] "Mean"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
    0.4756130     0.4529462     0.4153998     0.5363601     0.1190008
[1] "Standard Deviation"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
   0.04237295    0.04342732    0.04146497    0.04235445    0.01643218
[1] "Variance"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
 0.0017954667  0.0018859319  0.0017193438  0.0017938991  0.0002700165
```

	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1
Class: 1	0.9801587	0.9920597	0.8133705	0.9994347	0.8133705	0.9801587	0.8849942
Class: 2	NA	0.9797657	NA	NA	0.5094336	NA	NA
Class: 3	NA	0.9750424	NA	NA	0.3958956	NA	NA
Class: 4	0.3168735	0.9865459	0.7336897	0.9241201	0.7336897	0.3168735	0.4356494
Class: 5	NA	0.9668659	NA	NA	0.2133135	NA	NA
Class: 6	NA	0.9722488	NA	NA	0.2401778	NA	NA
Class: 7	0.2991682	0.9801867	0.5765348	0.9373213	0.5765348	0.2991682	0.3855665
Class: 8	NA	0.9682442	NA	NA	0.0000000	NA	NA
Class: 9	0.7869265	0.9913654	0.8041808	0.9854300	0.8041808	0.7869265	0.7784585
Class: 10	NA	0.9828449	NA	NA	0.4809091	NA	NA
Class: 11	NA	0.9717214	NA	NA	0.2702289	NA	NA
Class: 12	0.6180876	0.9870145	0.5850848	0.9877455	0.5850848	0.6180876	0.5859110
Class: 13	0.8264047	0.9886607	0.6925321	0.9899767	0.6925321	0.8264047	0.7175648
Class: 14	NA	0.9859796	NA	NA	0.6440426	NA	NA
Class: 15	NA	0.9659664	NA	NA	0.1200286	NA	NA
Class: 16	0.5610162	0.9840797	0.6401998	0.9763632	0.6401998	0.5610162	0.5807226
Class: 17	0.4295689	0.9841276	0.6748389	0.9588354	0.6748389	0.4295689	0.5086260
Class: 18	NA	0.9635175	NA	NA	0.1896798	NA	NA
Class: 19	NA	0.9678765	NA	NA	0.0000000	NA	NA
Class: 20	NA	0.9754002	NA	NA	0.4035426	NA	NaN
Class: 21	0.4099140	0.9843252	0.5777411	0.9729772	0.5777411	0.4099140	0.4623835
Class: 22	0.5910761	0.9892292	0.7428784	0.9773510	0.7428784	0.5910761	0.6496129
Class: 23	0.4364493	0.9819673	0.5476598	0.9602258	0.5476598	0.4364493	0.4440366
Class: 24	0.3023846	0.9743842	0.4233866	0.9441814	0.4233866	0.3023846	NaN
Class: 25	NA	0.9777931	NA	NA	0.4796082	NA	NA
Class: 26	0.5053931	0.9816633	0.4907506	0.9791754	0.4907506	0.5053931	NaN

```
         Prevalence Detection Rate Detection Prevalence Balanced Accuracy
Class: 1 0.03358065    0.033033208          0.04070078         0.9861092
Class: 2 0.05656934    0.018613139          0.03796757                NA
Class: 3 0.02098741    0.016060399          0.04051696                NA
Class: 4 0.10567699    0.033217026          0.04526082         0.6517097
Class: 5 0.01715328    0.008211679          0.04088460                NA
Class: 6 0.02080292    0.010218978          0.03741611                NA
Class: 7 0.08559196    0.025731264          0.04380030         0.6396774
Class: 8 0.00000000    0.000000000          0.03175584                NA
Class: 9 0.04617991    0.032305286          0.04051964         0.8891460
Class: 10 0.01824818   0.014781022          0.03176051                NA
Class: 11 0.03595893   0.010955589          0.03851234                NA
Class: 12 0.03030267   0.018437341          0.03103126         0.8025510
Class: 13 0.03413278   0.024453892          0.03540480         0.9075327
Class: 14 0.03120438   0.019708029          0.03339951                NA
Class: 15 0.01643673   0.005475789          0.03905577                NA
Class: 16 0.04891246   0.026280046          0.04143004         0.7725480
Class: 17 0.06680837   0.027376942          0.04215997         0.7068483
Class: 18 0.01551095   0.008211679          0.04416526                NA
Class: 19 0.00000000   0.000000000          0.03212347                NA
Class: 20 0.02464373   0.015331809          0.03942274                NA
Class: 21 0.04453691   0.018433999          0.03339951         0.6971196
Class: 22 0.05311623   0.031393546          0.04161386         0.7901526
Class: 23 0.05894428   0.020623112          0.03759859         0.7092083
Class: 24 0.07301543   0.019711371          0.04361982         0.6383844
Class: 25 0.02408759   0.019525547          0.04125224                NA
Class: 26 0.03759793   0.017522259          0.03522767         0.7435282
```

Cross Validation – SVM

Let's do Cross validation using SVM.

The accuracy of the model with CV is 0.9376 and variance is in the power of e-05. CV on SVM show slight drop in accuracy for training data set. Without CV the model accuracy was 0.95862 and with CV the model's accuracy is 0.9376.

```
> crossValidationModels(trdf, "svm", 20, "train")
[1] "varianceEstimation"
                          [,1]
Mean of Accuracies     9.377e-01
Variance of Accuracies 4.814e-05
[1] "Mean"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
    0.9376505     0.9350452     0.9160457     0.9550865     0.0559342
[1] "Standard Deviation"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
  0.006938564   0.007223598   0.007846239   0.005955631   0.003734078
[1] "Variance"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
 4.814368e-05  5.218037e-05  6.156347e-05  3.546954e-05  1.394334e-05
         Sensitivity Specificity Pos Pred Value Neg Pred Value Precision    Recall        F1
Class: 1   0.9835545   0.9991034      0.9783767      0.9993498 0.9783767 0.9835545 0.9806763
Class: 2   0.8669476   0.9979520      0.9524688      0.9936308 0.9524688 0.8669476 0.9061062
Class: 3   0.9703603   0.9968246      0.9231370      0.9988587 0.9231370 0.9703603 0.9455377
Class: 4   0.9026513   0.9974590      0.9431167      0.9955065 0.9431167 0.9026513 0.9214032
Class: 5   0.8982535   0.9970689      0.9260832      0.9958522 0.9260832 0.8982535 0.9100373
Class: 6   0.9018317   0.9975391      0.9468216      0.9955086 0.9468216 0.9018317 0.9220933
Class: 7   0.8995822   0.9976311      0.9418438      0.9956855 0.9418438 0.8995822 0.9192985
Class: 8   0.8632162   0.9928461      0.7719423      0.9962095 0.7719423 0.8632162 0.8091827
Class: 9   0.9820057   0.9969868      0.9313560      0.9992660 0.9313560 0.9820057 0.9551662
Class: 10  0.9486801   0.9979722      0.9451413      0.9980516 0.9451413 0.9486801 0.9459396
Class: 11  0.9250262   0.9963412      0.9115270      0.9970729 0.9115270 0.9250262 0.9163760
Class: 12  0.9980769   0.9971775      0.9156455      0.9999183 0.9156455 0.9980769 0.9541164
Class: 13  0.9686945   0.9984682      0.9527395      0.9988664 0.9527395 0.9686945 0.9593914
Class: 14  0.9288798   0.9981377      0.9445719      0.9975703 0.9445719 0.9288798 0.9356286
Class: 15  0.9019054   0.9971351      0.9357389      0.9955931 0.9357389 0.9019054 0.9168381
Class: 16  0.9862509   0.9955333      0.8959750      0.9994276 0.8959750 0.9862509 0.9375895
Class: 17  0.9516360   0.9981264      0.9547972      0.9979637 0.9547972 0.9516360 0.9523176
Class: 18  0.8384165   0.9966517      0.9163260      0.9926655 0.9163260 0.8384165 0.8739527
Class: 19  0.9601983   0.9983689      0.9625487      0.9983703 0.9625487 0.9601983 0.9606491
Class: 20  0.9715999   0.9980507      0.9522611      0.9988606 0.9522611 0.9715999 0.9611414
Class: 21  0.9601070   0.9982983      0.9560871      0.9986234 0.9560871 0.9601070 0.9563573
Class: 22  0.9671313   0.9975511      0.9477003      0.9987003 0.9477003 0.9671313 0.9557560
Class: 23  0.9510941   0.9984622      0.9565897      0.9982167 0.9565897 0.9510941 0.9527953
Class: 24  0.9435467   0.9976293      0.9474817      0.9975540 0.9474817 0.9435467 0.9445126
Class: 25  0.9707091   0.9992644      0.9829999      0.9986945 0.9829999 0.9707091 0.9762843
Class: 26  0.9749608   0.9986284      0.9594159      0.9991100 0.9594159 0.9749608 0.9659450
```

```
         Prevalence Detection Rate Detection Prevalence Balanced Accuracy
Class: 1 0.04013216     0.03950655           0.04036691         0.9913289
Class: 2 0.04662534     0.04052340           0.04247897         0.9324498
Class: 3 0.03637568     0.03528071           0.03833187         0.9835925
Class: 4 0.04466879     0.04036581           0.04279098         0.9500551
Class: 5 0.03848787     0.03449812           0.03731392         0.9476612
Class: 6 0.04396518     0.03966231           0.04200924         0.9496854
Class: 7 0.04185153     0.03770540           0.03997433         0.9486067
Class: 8 0.02792669     0.02425005           0.03121283         0.9280312
Class: 9 0.03942953     0.03872530           0.04162021         0.9894963
Class: 10 0.03661055    0.03473286           0.03668843         0.9733262
Class: 11 0.03864498    0.03582844           0.03934871         0.9606837
Class: 12 0.02988385    0.02980560           0.03254365         0.9976272
Class: 13 0.03332490    0.03222968           0.03371601         0.9835813
Class: 14 0.03348200    0.03113532           0.03293464         0.9635087
Class: 15 0.04255746    0.03833272           0.04107101         0.9495203
Class: 16 0.03856600    0.03801839           0.04232113         0.9908921
Class: 17 0.03958419    0.03762825           0.03942745         0.9748812
Class: 18 0.04302646    0.03598616           0.03919356         0.9175341
Class: 19 0.04138314    0.03981832           0.04138302         0.9792836
Class: 20 0.03911458    0.03801949           0.03989669         0.9848253
Class: 21 0.03528071    0.03395088           0.03559358         0.9792027
Class: 22 0.03903609    0.03778438           0.04013119         0.9823412
Class: 23 0.03520308    0.03348200           0.03496833         0.9747782
Class: 24 0.04161654    0.03926961           0.04153829         0.9705880
Class: 25 0.04161764    0.04036605           0.04107028         0.9849867
Class: 26 0.03160505    0.03074469           0.03207478         0.9867946
```

Let's see next the behavior of CV on testing dataset.
The accuracy of the model with CV is 0.03614 and variance is in the power of e-04 for the test data set. CV on SVM show high drop in accuracy for testing data set. Without CV the model accuracy was 0.95862 and with CV the model's accuracy is 0.9376. The model is overfitting this can be because of predictor interaction. Hence CV using SVM is not well suited for this data set.

```
> crossValidationModels(tstdf, "svm", 20, "test")
[1] "varianceEstimation"
                            [,1]
Mean of Accuracies      0.0361400
Variance of Accuracies 0.0001218
[1] "Mean"
     Accuracy          Kappa AccuracyLower AccuracyUpper  AccuracyNull
  0.036137403  -0.002885244   0.017728580   0.065349056   0.068807652
[1] "Standard Deviation"
     Accuracy          Kappa AccuracyLower AccuracyUpper  AccuracyNull
  0.011036330   0.011426623   0.007255937   0.014379055   0.007509555
[1] "Variance"
     Accuracy          Kappa AccuracyLower AccuracyUpper  AccuracyNull
 1.218006e-04  1.305677e-04  5.264862e-05  2.067572e-04  5.639341e-05
          Sensitivity Specificity Pos Pred Value Neg Pred Value  Precision     Recall  F1 Prevalence
Class: 1  0.04002040   0.9562047     0.03821123      0.9585794 0.03821123 0.04002040 NaN 0.04124823
Class: 2  0.04518239   0.9544329     0.03958495      0.9564450 0.03958495 0.04518239 NaN 0.04344269
Class: 3  0.04623034   0.9626423     0.04663947      0.9630041 0.04663947 0.04623034 NaN 0.03741411
Class: 4  0.02158991   0.9562435     0.02940476      0.9498531 0.02940476 0.02158991 NaN 0.04927609
Class: 5  0.02702076   0.9604168     0.02928807      0.9547614 0.02928807 0.02702076 NaN 0.04471739
Class: 6  0.03222174   0.9587572     0.03367244      0.9600270 0.03367244 0.03222174 NaN 0.03978904
Class: 7  0.06398846   0.9594254     0.05911859      0.9530653 0.05911859 0.06398846 NaN 0.04781690
Class: 8  0.03807720   0.9676060     0.02388889      0.9752780 0.02388889 0.03807720 NaN 0.02482220
Class: 9  0.04954060   0.9587809     0.04590999      0.9624262 0.04590999 0.04954060 NaN 0.03778108
Class: 10 0.04144841   0.9664033     0.03530229      0.9726083 0.03530229 0.04144841 NaN 0.02756277
Class: 11 0.01535876   0.9623020     0.02109002      0.9611928 0.02109002 0.01535876 NaN 0.03814738
Class: 12 0.02186508   0.9682751     0.02097222      0.9724682 0.02097222 0.02186508 NaN 0.02738162
Class: 13 0.04147653   0.9659244     0.03695378      0.9648437 0.03695378 0.04147653 NaN 0.03540347
Class: 14 0.03083333   0.9680924     0.02635101      0.9668241 0.02635101 0.03083333 NaN 0.03303454
Class: 15 0.03617711   0.9549361     0.03038096      0.9598746 0.03038096 0.03617711 NaN 0.03996818
Class: 16 0.05311688   0.9576864     0.03969262      0.9622604 0.03969262 0.05311688 NaN 0.03778041
Class: 17 0.03030449   0.9586020     0.02164558      0.9556946 0.02164558 0.03030449 NaN 0.04362049
Class: 18 0.03810472   0.9596446     0.03679945      0.9496357 0.03679945 0.03810472 NaN 0.05000602
Class: 19 0.02380952   0.9584480     0.01543040      0.9706671 0.01543040 0.02380952 NaN 0.02883680
Class: 20 0.02311383   0.9603740     0.01876128      0.9627539 0.01876128 0.02311383 NaN 0.03668418
Class: 21 0.04738095   0.9641840     0.05202381      0.9678337 0.05202381 0.04738095 NaN 0.03266891
Class: 22 0.05248397   0.9612208     0.03284939      0.9603272 0.03284939 0.05248397 NaN 0.03960656
Class: 23 0.03519841   0.9675319     0.04798993      0.9618673 0.04798993 0.03519841 NaN 0.03832852
Class: 24 0.03386780   0.9596355     0.03869103      0.9532543 0.03869103 0.03386780 NaN 0.04654020
Class: 25 0.03682958   0.9603283     0.04448163      0.9570325 0.04448163 0.03682958 NaN 0.04307706
Class: 26 0.01842949   0.9689627     0.02145022      0.9645841 0.02145022 0.01842949 NaN 0.03504519
```

```
         Detection Rate Detection Prevalence Balanced Accuracy
Class: 1   0.0016430042          0.04362116         0.4981126
Class: 2   0.0018248175          0.04544598         0.4998077
Class: 3   0.0018248175          0.03777773         0.5044363
Class: 4   0.0012773723          0.04288923         0.4889167
Class: 5   0.0012787091          0.03905845         0.4937188
Class: 6   0.0014605224          0.04106441         0.4954895
Class: 7   0.0027378947          0.04143405         0.5117069
Class: 8   0.0009124088          0.03248710         0.5028416
Class: 9   0.0018248175          0.04142937         0.5041607
Class: 10  0.0010948905          0.03376648         0.5039259
Class: 11  0.0007299270          0.03704914         0.4888304
Class: 12  0.0007299270          0.03157670         0.4950701
Class: 13  0.0014598540          0.03431259         0.5037005
Class: 14  0.0009124088          0.03175918         0.4994628
Class: 15  0.0016423358          0.04489920         0.4955566
Class: 16  0.0016430042          0.04234713         0.5054016
Class: 17  0.0010948905          0.04069744         0.4944532
Class: 18  0.0016430042          0.03996885         0.4988746
Class: 19  0.0007299270          0.04106107         0.4911288
Class: 20  0.0009124088          0.03905845         0.4917439
Class: 21  0.0016423358          0.03632657         0.5057825
Class: 22  0.0014598540          0.03869348         0.5068524
Class: 23  0.0014598540          0.03267025         0.5013652
Class: 24  0.0016430042          0.04015200         0.4967517
Class: 25  0.0018254859          0.03978971         0.4985789
Class: 26  0.0007299270          0.03066429         0.4936961
```

Cross Validation – KNN

As we saw from Assignment 2 from the Error Rate v/s k plot, value of K=1 gives the least error rate. Hence for Cross Validation also we will use k=1. From Cross Validation we can see the Accuracy of the KNN model with CV is 0.9595 and accuracy of the model without CV is 0.954197. The accuracy of the model has slightly increased for Cross Validation but there is no major difference between the accuracy and variance of the model with CV and without CV.

```
> crossValidationModels(dataset_without_outliers, "knn", 20, "train")
[1] "varianceEstimation"
                            [,1]
Mean of Accuracies      9.595e-01
Variance of Accuracies 2.211e-05
[1] "Mean"
     Accuracy          Kappa AccuracyLower AccuracyUpper  AccuracyNull
   0.95948079     0.95780622     0.94460160     0.97129041     0.05245566
[1] "Standard Deviation"
     Accuracy          Kappa AccuracyLower AccuracyUpper  AccuracyNull
  0.004701750   0.004895427   0.005394051   0.003969312   0.003133673
[1] "Variance"
     Accuracy          Kappa AccuracyLower AccuracyUpper  AccuracyNull
 2.210645e-05  2.396521e-05  2.909579e-05  1.575544e-05  9.819908e-06
         Sensitivity Specificity Pos Pred Value Neg Pred Value Precision    Recall        F1
Class: 1   0.9975595   0.9998860      0.9972470      0.9998855 0.9972470 0.9975595 0.9973686
Class: 2   0.9224440   0.9976007      0.9429430      0.9964580 0.9429430 0.9224440 0.9310448
Class: 3   0.9681907   0.9989740      0.9739172      0.9988071 0.9739172 0.9681907 0.9706782
Class: 4   0.9470431   0.9985672      0.9689268      0.9975403 0.9689268 0.9470431 0.9572400
Class: 5   0.9366186   0.9975494      0.9415467      0.9974956 0.9415467 0.9366186 0.9380298
Class: 6   0.9436515   0.9968650      0.9258711      0.9976615 0.9258711 0.9436515 0.9336045
Class: 7   0.9605827   0.9984008      0.9624641      0.9982864 0.9624641 0.9605827 0.9610756
Class: 8   0.8850416   0.9958152      0.8773171      0.9963270 0.8773171 0.8850416 0.8787768
Class: 9   0.9630656   0.9984013      0.9632332      0.9984018 0.9632332 0.9630656 0.9624686
Class: 10  0.9473664   0.9978996      0.9441456      0.9980719 0.9441456 0.9473664 0.9444881
Class: 11  0.9340743   0.9966999      0.9172016      0.9973784 0.9172016 0.9340743 0.9246859
Class: 12  0.9797897   0.9992068      0.9773165      0.9993789 0.9773165 0.9797897 0.9781983
Class: 13  0.9876119   0.9994338      0.9839441      0.9995463 0.9839441 0.9876119 0.9854798
Class: 14  0.9645002   0.9987520      0.9657617      0.9988117 0.9657617 0.9645002 0.9646090
Class: 15  0.9513405   0.9987404      0.9713909      0.9978286 0.9713909 0.9513405 0.9610120
Class: 16  0.9493267   0.9974818      0.9442018      0.9977700 0.9442018 0.9493267 0.9462120
Class: 17  0.9708855   0.9982880      0.9594827      0.9988014 0.9594827 0.9708855 0.9647756
Class: 18  0.9326032   0.9971435      0.9344040      0.9971463 0.9344040 0.9326032 0.9324360
Class: 19  0.9805874   0.9993744      0.9843346      0.9992011 0.9843346 0.9805874 0.9821609
Class: 20  0.9674502   0.9987435      0.9704514      0.9986911 0.9704514 0.9674502 0.9684697
Class: 21  0.9825056   0.9993181      0.9821733      0.9993765 0.9821733 0.9825056 0.9820428
Class: 22  0.9598426   0.9984566      0.9650134      0.9983472 0.9650134 0.9598426 0.9618972
Class: 23  0.9851154   0.9994889      0.9860530      0.9994304 0.9860530 0.9851154 0.9854440
Class: 24  0.9760312   0.9984544      0.9669529      0.9989725 0.9669529 0.9760312 0.9710378
Class: 25  0.9733671   0.9989724      0.9761153      0.9987991 0.9761153 0.9733671 0.9743964
Class: 26  0.9829105   0.9993209      0.9801980      0.9994344 0.9801980 0.9829105 0.9811440
```

```
          Prevalence Detection Rate Detection Prevalence Balanced Accuracy
Class: 1  0.04046474     0.04035527           0.04046474         0.9987228
Class: 2  0.04221649     0.03882187           0.04112168         0.9600224
Class: 3  0.03914926     0.03799926           0.03898484         0.9835824
Class: 4  0.04451708     0.04216244           0.04353138         0.9728051
Class: 5  0.03843738     0.03602834           0.03838273         0.9670840
Class: 6  0.03986245     0.03761723           0.04062868         0.9702583
Class: 7  0.04123097     0.03958845           0.04112174         0.9794918
Class: 8  0.03088203     0.02732294           0.03137491         0.9404284
Class: 9  0.04128681     0.03975358           0.04128663         0.9807335
Class: 10 0.03504330     0.03318166           0.03520777         0.9726330
Class: 11 0.03843810     0.03591947           0.03909509         0.9653871
Class: 12 0.03192262     0.03132033           0.03208703         0.9894982
Class: 13 0.03411332     0.03367526           0.03422273         0.9935229
Class: 14 0.03301761     0.03186779           0.03307237         0.9816261
Class: 15 0.04134056     0.03925968           0.04046450         0.9750404
Class: 16 0.04177843     0.03964292           0.04205243         0.9734042
Class: 17 0.03975268     0.03860269           0.04024545         0.9845868
Class: 18 0.04068266     0.03794498           0.04068242         0.9648733
Class: 19 0.03876758     0.03800088           0.03860311         0.9899809
Class: 20 0.03975149     0.03849208           0.03969678         0.9830968
Class: 21 0.03487882     0.03427647           0.03493365         0.9909118
Class: 22 0.04068410     0.03909635           0.04057487         0.9791496
Class: 23 0.03580952     0.03526193           0.03575476         0.9923022
Class: 24 0.04172325     0.04073761           0.04221577         0.9872428
Class: 25 0.04128567     0.04013574           0.04112132         0.9861698
Class: 26 0.03296308     0.03241556           0.03307261         0.9911157
```

Both bias and variance are reduced by Cross Validation.

Bagging

Bagging, also known as Bootstrap aggregating, is an ensemble learning technique that helps to improve the performance and accuracy of machine learning algorithms. It is used to deal with bias-variance trade-offs and **reduces the variance of a prediction model**. I have created a method which accepts different model and perform bagging on the models as shown below. I have created a function named as 'baggingModels' which accepts dataset and type as the parameters.

```
> ##############################################################################
> #                                 Bagging                                    #
> ##############################################################################
> trdf <- trainingDataset_withoutOutliers
> tstdf <- testingDataset_withoutOutliers
>
> baggingModels=function(df,type){
+   # For lda
+   if(type=="lda"){
+     runModel<-function(df) {
+       lda(Letter~.,data=df[sample(1:nrow(df),nrow(df),replace=T),])
+     }
+   }
+   # For qda
+   else if(type=="qda"){
+     runModel<-function(df) {
+       qda(Letter~.,data=df[sample(1:nrow(df),nrow(df),replace=T),])
+     }
+   }
+   # For SVM
+   else if(type=="svm"){
+     runModel<-function(df) {
+       svm(Letter~.,data=df[sample(1:nrow(df),nrow(df),replace=T),])
+     }
+     lapplyrunmodel <- function(x) runModel(df)
+     models<-lapply(1:5,lapplyrunmodel)
+     bagging_preds <- lapply(models,FUN=function(M,D=df[,-c(which(names(df)=="Letter"))]){predict(M,D)})
+     bagging_cfm <-lapply(bagging_preds,FUN=function(P,A=df[[c(which(names(df)=="Letter"))]]){
+       pred_tbl<-table(A,P)
+       pred_cfm<-caret::confusionMatrix(pred_tbl)
+       pred_cfm
+     })
+   }
+   # For Tree
+   else if(type=="tree"){
+     runModel<-function(df) {
+       rpart(Letter~.,data=df[sample(1:nrow(df),nrow(df),replace=T),])
+     }
+     lapplyrunmodel <- function(x) runModel(df)
+     models<-lapply(1:100,lapplyrunmodel)
```

```
+     bagging_preds <- lapply(models,FUN=function(M,D=df[,-c(which(names(df)=="Letter"))])predict(M,D,type
="class"))
+     bagging_cfm <-lapply(bagging_preds,FUN=function(P,A=df[[c(which(names(df)=="Letter"))]]){
+       pred_tbl<-table(A,P)
+       pred_cfm<-caret::confusionMatrix(pred_tbl)
+       pred_cfm
+     })
+   }
+   else {
+     print("type should be 'lda' 'qda' 'tree' 'svm'")
+     return()
+   }
+
+   if(type !="tree" && type !="svm")
+   {
+     lapplyrunmodel <- function(x) runModel(df)
+     models<-lapply(1:100,lapplyrunmodel)
+     bagging_preds <- lapply(models,FUN=
+                               function(M,D=df[,-c(which(names(df)=="Letter"))])
+                                 predict(M,D))
+     bagging_cfm <-lapply(bagging_preds,FUN=function(P,A=df[[c(which(names(df)=="Letter"))]]){
+       pred_tbl<-table(A,P$class)
+       pred_cfm<-caret::confusionMatrix(pred_tbl)
+       pred_cfm
+     })
+   }
+
+   bagging.perf<-as.data.frame(do.call('rbind',lapply(bagging_cfm,FUN=function(cfm)c(cfm$overall))))
+   acc_varEstp <- bagging.perf$Accuracy
+   mean_varEstp = signif(mean(acc_varEstp),4)
+   var_varEstp = signif(var(acc_varEstp),4)
+   varEstp = data.frame(mean_varEstp,var_varEstp)
+   names(varEstp) = c("Mean of Accuracies","Variance of Accuracies")
+   varianceEstimation <- t(varEstp)
+   print("varianceEstimation")
+   print(varianceEstimation)
+
+   bagging.perf.mean<-apply(bagging.perf[bagging.perf$AccuracyPValue<0.01,-c(6:7)],2,mean)
+   print("Mean")
+   print(bagging.perf.mean)

+   bagging.perf.var<-apply(bagging.perf[bagging.perf$AccuracyPValue<0.01,-c(6:7)],2,sd)
+   print("Standard Deviation")
+   print(bagging.perf.var)
+   for(c in bagging_cfm) {
+     confusionMatrixByClass <- c$byClass
+     confusion_matrix_sd <- c$byClass
+     break
+   }
+   y <- 0
+   z <- c()
+   cfm_len <- length(bagging_cfm)
+   for(column in colnames(confusionMatrixByClass)) {
+     for(row in rownames(confusionMatrixByClass)) {
+       for (x in bagging_cfm) {
+         my_mat <- as.matrix(x$byClass)
+         y <- y + my_mat[row,column]
+         z <- c(z,my_mat[row,column])
+       }
+       confusionMatrixByClass[row,column] <- y/cfm_len
+       confusion_matrix_sd[row, column] <- sd(z)
+       y <- 0
+       z <- c()
+     }
+   }
+   confusionMatrixByClass
+ }
```

Bagging – LDA

Let's do Bagging using LDA.

The accuracy of the model with Bagging is 0.7184 and variance is in the power of e-06. Bagging on LDA show slight drop in accuracy for training data set. Without Bagging the model accuracy was 0.7219.

```
> baggingModels(trdf, "lda")
[1] "varianceEstimation"
                          [,1]
Mean of Accuracies     7.184e-01
Variance of Accuracies 5.564e-06
[1] "Mean"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
   0.71837922    0.70699967    0.71049420    0.72616472    0.05706352
[1] "Standard Deviation"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
  0.002358823   0.002453837   0.002378139   0.002338430   0.001683411
          Sensitivity Specificity Pos Pred Value Neg Pred Value Precision    Recall        F1
Class: 1    0.9031966   0.9942163      0.8622287      0.9961086 0.8622287 0.9031966 0.8822058
Class: 2    0.5682783   0.9893201      0.7628913      0.9742578 0.7628913 0.5682783 0.6512182
Class: 3    0.7082590   0.9903307      0.7580816      0.9875403 0.7580816 0.7082590 0.7322529
Class: 4    0.7249262   0.9895001      0.7656856      0.9869911 0.7656856 0.7249262 0.7445880
Class: 5    0.6677145   0.9822857      0.5395388      0.9895580 0.5395388 0.6677145 0.5963538
Class: 6    0.6781847   0.9867625      0.6984916      0.9854266 0.6984916 0.6781847 0.6879366
Class: 7    0.6175264   0.9802732      0.5232094      0.9864426 0.5232094 0.6175264 0.5660531
Class: 8    0.3621724   0.9818689      0.4412030      0.9748631 0.4412030 0.3621724 0.3973980
Class: 9    0.8717975   0.9918599      0.8119737      0.9948098 0.8119737 0.8717975 0.8407784
Class: 10   0.8466406   0.9882920      0.6904051      0.9952270 0.6904051 0.8466406 0.7604491
Class: 11   0.6849976   0.9863982      0.6675547      0.9873862 0.6675547 0.6849976 0.6758694
Class: 12   0.8841169   0.9908028      0.7249038      0.9967990 0.7249038 0.8841169 0.7965882
Class: 13   0.8947792   0.9962918      0.8937123      0.9963248 0.8937123 0.8947792 0.8941637
Class: 14   0.7350024   0.9925194      0.7807838      0.9903931 0.7807838 0.7350024 0.7570139
Class: 15   0.6087277   0.9899707      0.7684381      0.9788221 0.7684381 0.6087277 0.6792017
Class: 16   0.8546288   0.9891656      0.7535305      0.9943257 0.7535305 0.8546288 0.8008122
Class: 17   0.6689763   0.9854737      0.6455556      0.9868607 0.6455556 0.6689763 0.6568606
Class: 18   0.6500641   0.9894124      0.7419162      0.9836677 0.7419162 0.6500641 0.6927256
Class: 19   0.4792113   0.9806449      0.5546503      0.9739445 0.5546503 0.4792113 0.5139622
Class: 20   0.8361582   0.9884370      0.7200980      0.9940891 0.7200980 0.8361582 0.7733182
Class: 21   0.8128988   0.9922420      0.7895604      0.9932776 0.7895604 0.8128988 0.8009464
Class: 22   0.8219386   0.9937207      0.8500000      0.9922191 0.8500000 0.8219386 0.8351765
Class: 23   0.8370481   0.9960489      0.8912081      0.9937035 0.8912081 0.8370481 0.8632131
Class: 24   0.6901606   0.9891525      0.7506403      0.9853522 0.7506403 0.6901606 0.7189052
Class: 25   0.8543502   0.9817586      0.5678857      0.9958382 0.5678857 0.8543502 0.6815615
Class: 26   0.7661613   0.9904246      0.7110462      0.9927398 0.7110462 0.7661613 0.7371352
          Prevalence Detection Rate Detection Prevalence Balanced Accuracy
Class: 1  0.03853645     0.03480210           0.04036295         0.9487065
Class: 2  0.05705257     0.03240379           0.04247497         0.7787992
Class: 3  0.04103880     0.02905663           0.03832916         0.8492948
Class: 4  0.04521433     0.03276205           0.04278786         0.8572131
Class: 5  0.03018382     0.02013141           0.03731227         0.8250001
Class: 6  0.04330178     0.02934058           0.04200563         0.8324736
Class: 7  0.03392913     0.02091364           0.03997184         0.7988998
Class: 8  0.03812265     0.01377034           0.03121089         0.6720206
Class: 9  0.03876408     0.03378989           0.04161452         0.9318287
Class: 10 0.02992647     0.02532854           0.03668648         0.9174663
Class: 11 0.03838314     0.02626564           0.03934606         0.8356979
Class: 12 0.02668570     0.02358886           0.03254068         0.9374599
Class: 13 0.03368195     0.03013063           0.03371402         0.9455355
Class: 14 0.03500313     0.02571261           0.03293179         0.8637609
Class: 15 0.05186561     0.03155742           0.04106696         0.7993492
Class: 16 0.03732243     0.03188830           0.04231852         0.9218972
Class: 17 0.03807181     0.02545056           0.03942428         0.8272250
Class: 18 0.04476768     0.02907541           0.03918961         0.8197382
Class: 19 0.04792866     0.02295135           0.04137985         0.7299281
Class: 20 0.03440238     0.02872732           0.03989362         0.9122976
Class: 21 0.03458464     0.02810153           0.03559136         0.9025704
Class: 22 0.04157775     0.03410904           0.04012829         0.9078296
Class: 23 0.03723795     0.03116161           0.03496558         0.9165485
Class: 24 0.04521824     0.03117882           0.04153630         0.8396565
Class: 25 0.02731227     0.02332134           0.04106696         0.9180544
Class: 26 0.02988658     0.02285982           0.03214956         0.8782929
```

Let's see next the behavior of LDA on testing dataset.

Bagging Test– LDA

The accuracy of bagging with LDA is 0.7189 shows slight drop and without bagging the model accuracy is 0.72281.

```
> baggingModels(trdf, "lda")
[1] "varianceEstimation"
                          [,1]
Mean of Accuracies     7.189e-01
Variance of Accuracies 5.249e-06
[1] "Mean"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
   0.71886264    0.70750142    0.71098157    0.72664396    0.05687265
[1] "Standard Deviation"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
  0.002291107   0.002383502   0.002309878   0.002271289   0.001619462
[1] "Variance"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
 5.249169e-06  5.681081e-06  5.335537e-06  5.158754e-06  2.622657e-06
          Sensitivity Specificity Pos Pred Value Neg Pred Value Precision    Recall        F1
Class: 1    0.9060386   0.9942462      0.8629264      0.9962317 0.8629264 0.9060386 0.8839230
Class: 2    0.5702117   0.9893118      0.7626519      0.9744678 0.7626519 0.5702117 0.6523916
Class: 3    0.7086329   0.9904050      0.7599592      0.9875281 0.7599592 0.7086329 0.7332997
Class: 4    0.7231863   0.9895972      0.7679159      0.9868350 0.7679159 0.7231863 0.7447185
Class: 5    0.6714261   0.9821146      0.5348847      0.9898286 0.5348847 0.6714261 0.5950019
Class: 6    0.6796367   0.9867581      0.6983613      0.9855344 0.6983613 0.6796367 0.6886668
Class: 7    0.6146204   0.9804886      0.5286693      0.9861395 0.5286693 0.6146204 0.5680876
Class: 8    0.3629574   0.9818101      0.4392231      0.9750892 0.4392231 0.3629574 0.3971151
Class: 9    0.8706979   0.9918555      0.8118797      0.9947568 0.8118797 0.8706979 0.8401914
Class: 10   0.8445752   0.9883372      0.6916418      0.9951433 0.6916418 0.8445752 0.7603611
Class: 11   0.6814714   0.9864616      0.6692048      0.9871444 0.6692048 0.6814714 0.6749754
Class: 12   0.8825347   0.9908198      0.7254327      0.9967473 0.7254327 0.8825347 0.7962614
Class: 13   0.8942885   0.9962507      0.8925290      0.9963102 0.8925290 0.8942885 0.8933210
Class: 14   0.7382126   0.9925069      0.7803800      0.9905557 0.7803800 0.7382126 0.7585263
Class: 15   0.6073746   0.9900092      0.7693714      0.9786704 0.7693714 0.6073746 0.6786997
Class: 16   0.8547645   0.9892211      0.7548059      0.9943225 0.7548059 0.8547645 0.8015893
Class: 17   0.6712040   0.9854176      0.6441071      0.9870261 0.6441071 0.6712040 0.6571930
Class: 18   0.6503735   0.9894046      0.7417166      0.9837051 0.7417166 0.6503735 0.6928642
Class: 19   0.4811631   0.9806719      0.5551985      0.9741281 0.5551985 0.4811631 0.5153263
Class: 20   0.8400454   0.9884141      0.7194902      0.9942741 0.7194902 0.8400454 0.7747593
Class: 21   0.8110418   0.9922542      0.7899121      0.9931933 0.7899121 0.8110418 0.8002336
Class: 22   0.8216103   0.9937617      0.8509942      0.9922133 0.8509942 0.8216103 0.8356397
Class: 23   0.8358671   0.9960495      0.8912304      0.9936492 0.8912304 0.8358671 0.8625958
Class: 24   0.6886830   0.9891958      0.7516761      0.9852330 0.7516761 0.6886830 0.7185843
Class: 25   0.8587820   0.9818972      0.5711810      0.9959630 0.5711810 0.8587820 0.6854162
Class: 26   0.7668708   0.9904176      0.7108273      0.9927794 0.7108273 0.7668708 0.7374499
          Prevalence Detection Rate Detection Prevalence Balanced Accuracy
Class: 1  0.03844650     0.03483026           0.04036295         0.9501424
Class: 2  0.05684136     0.03239362           0.04247497         0.7797617
Class: 3  0.04112250     0.02912860           0.03832916         0.8495190
Class: 4  0.04545917     0.03285748           0.04278786         0.8563918
Class: 5  0.02974969     0.01995776           0.03731227         0.8267704
Class: 6  0.04319305     0.02933511           0.04200563         0.8331974
Class: 7  0.03443836     0.02113188           0.03997184         0.7975545
Class: 8  0.03784183     0.01370854           0.03121089         0.6723838
Class: 9  0.03881101     0.03378598           0.04161452         0.9312767
Class: 10 0.03005241     0.02537390           0.03668648         0.9164562
Class: 11 0.03868038     0.02633057           0.03934606         0.8339665
Class: 12 0.02675297     0.02360607           0.03254068         0.9366773
Class: 13 0.03365613     0.03009074           0.03371402         0.9452696
Class: 14 0.03483260     0.02569931           0.03293179         0.8653598
Class: 15 0.05204944     0.03159574           0.04106696         0.7986919
Class: 16 0.03737954     0.03194227           0.04231852         0.9219928
Class: 17 0.03785591     0.02539346           0.03942428         0.8283108
Class: 18 0.04472387     0.02906758           0.03918961         0.8198891
Class: 19 0.04777534     0.02297403           0.04137985         0.7309175
Class: 20 0.03420056     0.02870307           0.03989362         0.9142297
Class: 21 0.03467850     0.02811405           0.03559136         0.9016480
Class: 22 0.04162312     0.03414894           0.04012829         0.9076860
Class: 23 0.03729115     0.03116239           0.03496558         0.9159583
Class: 24 0.04537547     0.03122184           0.04153630         0.8389394
Class: 25 0.02732791     0.02345666           0.04106696         0.9203396
Class: 26 0.02984121     0.02285278           0.03214956         0.8786442
```

Bagging – QDA

The accuracy of the model with Bagging is 0.8930 and variance is in the power of e-06. Bagging on QDA show slight drop in accuracy for training data set. Without Bagging the model accuracy was 0.9.

```
> baggingModels(trdf, "qda")
[1] "varianceEstimation"
                            [,1]
Mean of Accuracies     8.930e-01
Variance of Accuracies 4.097e-06
[1] "Mean"
    Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
   0.8930358     0.8887196     0.8875497     0.8983423     0.0459332
[1] "Standard Deviation"
    Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
 0.0020240215  0.0021055674  0.0020681651  0.0019789385  0.0007994445
          Sensitivity Specificity Pos Pred Value Neg Pred Value Precision    Recall        F1
Class: 1    0.9620870   0.9977016      0.9453101      0.9984301 0.9453101 0.9620870 0.9535815
Class: 2    0.8783537   0.9956883      0.9029098      0.9944302 0.9029098 0.8783537 0.8902739
Class: 3    0.9128467   0.9961713      0.9038980      0.9965487 0.9038980 0.9128467 0.9082473
Class: 4    0.8606712   0.9962508      0.9163620      0.9933562 0.9163620 0.8606712 0.8875421
Class: 5    0.8735552   0.9948778      0.8677987      0.9951174 0.8677987 0.8735552 0.8705109
Class: 6    0.8510487   0.9954817      0.8971881      0.9930963 0.8971881 0.8510487 0.8733531
Class: 7    0.8726672   0.9944986      0.8678278      0.9947120 0.8678278 0.8726672 0.8700741
Class: 8    0.6956304   0.9899518      0.6879950      0.9902479 0.6879950 0.6956304 0.6913881
Class: 9    0.9585808   0.9940500      0.8623684      0.9983782 0.8623684 0.9585808 0.9078678
Class: 10   0.9381853   0.9969567      0.9200213      0.9976784 0.9200213 0.9381853 0.9288628
Class: 11   0.8508748   0.9936203      0.8441750      0.9939101 0.8441750 0.8508748 0.8472518
Class: 12   0.9290585   0.9962638      0.8887500      0.9977054 0.8887500 0.9290585 0.9083075
Class: 13   0.9472529   0.9980736      0.9447796      0.9981583 0.9447796 0.9472529 0.9459572
Class: 14   0.9130234   0.9967029      0.9031354      0.9970573 0.9031354 0.9130234 0.9079076
Class: 15   0.8381292   0.9957570      0.9012381      0.9925361 0.9012381 0.8381292 0.8684618
Class: 16   0.9505425   0.9959665      0.9085397      0.9979066 0.9085397 0.9505425 0.9290221
Class: 17   0.9064152   0.9947818      0.8726587      0.9962956 0.8726587 0.9064152 0.8891392
Class: 18   0.8287052   0.9961496      0.9059481      0.9923300 0.9059481 0.8287052 0.8653767
Class: 19   0.8520933   0.9953045      0.8914367      0.9933015 0.8914367 0.8520933 0.8711976
Class: 20   0.9222964   0.9947786      0.8740588      0.9969350 0.8740588 0.9222964 0.8974611
Class: 21   0.9056101   0.9980665      0.9476923      0.9963420 0.9476923 0.9056101 0.9260633
Class: 22   0.8931900   0.9967857      0.9232164      0.9953720 0.9232164 0.8931900 0.9078552
Class: 23   0.9306349   0.9981954      0.9502237      0.9974256 0.9502237 0.9306349 0.9402110
Class: 24   0.9341619   0.9945848      0.8746893      0.9973247 0.8746893 0.9341619 0.9033753
Class: 25   0.9213475   0.9939522      0.8583619      0.9968554 0.8583619 0.9213475 0.8886532
Class: 26   0.9204090   0.9981598      0.9446472      0.9972796 0.9446472 0.9204090 0.9322987
          Prevalence Detection Rate Detection Prevalence Balanced Accuracy
Class: 1  0.03966208     0.03815551           0.04036295         0.9798943
Class: 2  0.04368429     0.03835106           0.04247497         0.9370210
Class: 3  0.03796464     0.03464565           0.03832916         0.9545090
Class: 4  0.04556868     0.03920917           0.04278786         0.9284610
Class: 5  0.03707994     0.03237954           0.03731227         0.9342165
Class: 6  0.04430069     0.03768695           0.04200563         0.9232652
Class: 7  0.03976533     0.03468867           0.03997184         0.9335829
Class: 8  0.03092068     0.02147293           0.03121089         0.8427911
Class: 9  0.03744133     0.03588705           0.04161452         0.9763154
Class: 10 0.03598874     0.03375235           0.03668648         0.9675710
Class: 11 0.03906524     0.03321496           0.03934606         0.9222476
Class: 12 0.03114049     0.02892053           0.03254068         0.9626611
Class: 13 0.03363188     0.03185232           0.03371402         0.9726633
Class: 14 0.03258761     0.02974186           0.03293179         0.9548632
Class: 15 0.04416849     0.03701111           0.04106696         0.9169431
Class: 16 0.04045291     0.03844806           0.04231852         0.9732545
Class: 17 0.03796230     0.03440394           0.03942428         0.9505985
Class: 18 0.04287312     0.03550375           0.03918961         0.9124274
Class: 19 0.04330882     0.03688752           0.04137985         0.9236989
Class: 20 0.03781211     0.03486937           0.03989362         0.9585375
Class: 21 0.03725751     0.03372966           0.03559136         0.9518383
Class: 22 0.04148936     0.03704709           0.04012829         0.9449879
Class: 23 0.03570948     0.03322513           0.03496558         0.9644152
Class: 24 0.03889549     0.03633135           0.04153630         0.9643733
Class: 25 0.03826580     0.03525031           0.04106696         0.9576499
Class: 26 0.03300297     0.03036999           0.03214956         0.9592844
```

Let's see next the behavior of Bagging on testing dataset.

Mohit Shankar Pursnani **Letter Recognition Dataset** mp5578@nyu.edu

Assignment - 3

Bagging Test – QDA

The accuracy of bagging with QDA for testing data set is 0.8998 shows slight drop as without bagging the model accuracy is 0.918796.

```
> baggingModels(tstdf, "qda")
[1] "varianceEstimation"
                          [,1]
Mean of Accuracies     8.998e-01
Variance of Accuracies 1.299e-05
[1] "Mean"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
   0.89979562    0.89573856    0.89154168    0.90762097    0.04954927
[1] "Standard Deviation"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
  0.003603845   0.003749268   0.003728775   0.003474951   0.001885637
          Sensitivity Specificity Pos Pred Value Neg Pred Value Precision    Recall        F1
Class: 1    0.9686751   0.9978717      0.9497758      0.9986875 0.9497758 0.9686751 0.9590057
Class: 2    0.9024387   0.9958974      0.8959615      0.9961457 0.8959615 0.9024387 0.8988275
Class: 3    0.8857095   0.9959260      0.9035586      0.9950437 0.9035586 0.8857095 0.8941621
Class: 4    0.9048505   0.9961657      0.9191532      0.9953918 0.9191532 0.9048505 0.9117190
Class: 5    0.8863766   0.9943232      0.8666518      0.9952264 0.8666518 0.8863766 0.8760380
Class: 6    0.8730243   0.9959513      0.8959024      0.9949005 0.8959024 0.8730243 0.8839441
Class: 7    0.8568025   0.9946293      0.8828750      0.9931775 0.8828750 0.8568025 0.8691604
Class: 8    0.7265119   0.9915973      0.7439655      0.9907181 0.7439655 0.7265119 0.7343841
Class: 9    0.9629340   0.9946946      0.8738288      0.9985660 0.8738288 0.9629340 0.9159582
Class: 10   0.9346528   0.9974011      0.9206897      0.9978553 0.9206897 0.9346528 0.9271872
Class: 11   0.8781166   0.9929429      0.8233175      0.9953824 0.8233175 0.8781166 0.8494372
Class: 12   0.8984769   0.9972871      0.9152941      0.9966704 0.9152941 0.8984769 0.9065185
Class: 13   0.9054476   0.9974934      0.9317526      0.9963772 0.9317526 0.9054476 0.9179374
Class: 14   0.9372035   0.9969224      0.9108197      0.9978667 0.9108197 0.9372035 0.9235467
Class: 15   0.8515784   0.9955700      0.8911682      0.9936517 0.8911682 0.8515784 0.8705561
Class: 16   0.9337056   0.9963525      0.9155066      0.9971769 0.9155066 0.9337056 0.9243447
Class: 17   0.9192364   0.9960968      0.9112554      0.9964603 0.9112554 0.9192364 0.9150313
Class: 18   0.8310214   0.9966134      0.9270661      0.9912257 0.9270661 0.8310214 0.8760128
Class: 19   0.8098470   0.9955144      0.8651136      0.9931863 0.8651136 0.8098470 0.8359788
Class: 20   0.9246174   0.9959867      0.9020833      0.9969624 0.9020833 0.9246174 0.9130065
Class: 21   0.9411576   0.9981784      0.9472678      0.9979347 0.9472678 0.9411576 0.9439470
Class: 22   0.9230732   0.9966024      0.9217105      0.9966241 0.9217105 0.9230732 0.9220399
Class: 23   0.9627587   0.9977327      0.9418932      0.9985590 0.9418932 0.9627587 0.9520188
Class: 24   0.9499448   0.9957560      0.9071250      0.9977958 0.9071250 0.9499448 0.9278205
Class: 25   0.9019917   0.9947824      0.8785398      0.9958698 0.8785398 0.9019917 0.8898010
Class: 26   0.9530390   0.9975095      0.9317098      0.9983147 0.9317098 0.9530390 0.9421075
          Prevalence Detection Rate Detection Prevalence Balanced Accuracy
Class: 1  0.03990876     0.03864964           0.04069343         0.9832734
Class: 2  0.03771533     0.03400730           0.03795620         0.9491681
Class: 3  0.04135949     0.03660401           0.04051095         0.9408178
Class: 4  0.04599635     0.04159672           0.04525547         0.9505081
Class: 5  0.04000365     0.03542518           0.04087591         0.9403499
Class: 6  0.03842336     0.03351460           0.03740876         0.9344878
Class: 7  0.04518978     0.03866606           0.04379562         0.9257159
Class: 8  0.03260949     0.02362226           0.03175182         0.8590546
Class: 9  0.03677555     0.03539964           0.04051095         0.9788143
Class: 10 0.03131022     0.02923358           0.03175182         0.9660270
Class: 11 0.03614051     0.03170073           0.03850365         0.9355298
Class: 12 0.03162044     0.02839416           0.03102190         0.9478820
Class: 13 0.03647993     0.03298540           0.03540146         0.9514705
Class: 14 0.03247810     0.03041606           0.03339416         0.9670630
Class: 15 0.04090146     0.03480109           0.03905109         0.9235742
Class: 16 0.04062956     0.03792336           0.04142336         0.9650290
Class: 17 0.04180292     0.03841241           0.04215328         0.9576666
Class: 18 0.04932664     0.04093978           0.04416058         0.9138174
Class: 19 0.03437956     0.02778467           0.03211679         0.9026807
Class: 20 0.03847445     0.03555657           0.03941606         0.9603020
Class: 21 0.03362956     0.03163321           0.03339416         0.9696680
Class: 22 0.04158394     0.03834854           0.04160584         0.9598378
Class: 23 0.03679380     0.03540693           0.03759124         0.9802457
Class: 24 0.04183577     0.03972810           0.04379562         0.9728504
Class: 25 0.04019161     0.03623175           0.04124088         0.9483870
Class: 26 0.03443978     0.03281387           0.03521898         0.9752742
```

Bagging Train – SVM

Let's do Bagging using SVM.

The accuracy of the model with Bagging is 0.9507 and variance is in the power of e-06. Bagging on SVM show slight increase in accuracy for training data set. Without Bagging the model accuracy was 0.9401.

```
> baggingModels(trdf, "svm")
[1] "varianceEstimation"
                           [,1]
Mean of Accuracies     9.507e-01
Variance of Accuracies 1.389e-06
[1] "Mean"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
    0.9507196     0.9487285     0.9468248     0.9544077     0.0446965
[1] "Standard Deviation"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
 0.0011785450  0.0012258434  0.0012209387  0.0011355784  0.0002320463
[1] "Variance"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
 1.388968e-06  1.502692e-06  1.490691e-06  1.289538e-06  5.384547e-08
          Sensitivity Specificity Pos Pred Value Neg Pred Value Precision    Recall        F1
Class: 1    0.9887744   0.9993318      0.9841085      0.9995272 0.9841085 0.9887744 0.9864127
Class: 2    0.9106589   0.9979044      0.9528545      0.9958500 0.9528545 0.9106589 0.9312563
Class: 3    0.9823631   0.9972242      0.9302041      0.9993330 0.9302041 0.9823631 0.9555287
Class: 4    0.9153493   0.9978878      0.9528336      0.9960611 0.9528336 0.9153493 0.9337138
Class: 5    0.9326531   0.9975945      0.9379455      0.9973674 0.9379455 0.9326531 0.9352218
Class: 6    0.9269772   0.9981360      0.9575419      0.9966849 0.9575419 0.9269772 0.9419251
Class: 7    0.9065907   0.9980078      0.9522505      0.9959097 0.9522505 0.9065907 0.9288218
Class: 8    0.8970853   0.9946192      0.8325815      0.9969156 0.8325815 0.8970853 0.8634820
Class: 9    0.9779255   0.9973928      0.9398496      0.9990695 0.9398496 0.9779255 0.9584269
Class: 10   0.9541717   0.9981817      0.9522388      0.9982460 0.9522388 0.9541717 0.9530305
Class: 11   0.9526108   0.9968773      0.9236581      0.9981109 0.9236581 0.9526108 0.9378416
Class: 12   0.9963595   0.9972743      0.9187500      0.9998868 0.9187500 0.9963595 0.9559760
Class: 13   0.9788649   0.9988025      0.9656613      0.9992714 0.9656613 0.9788649 0.9721993
Class: 14   0.9552505   0.9982696      0.9491686      0.9984793 0.9491686 0.9552505 0.9521175
Class: 15   0.9111172   0.9982186      0.9584762      0.9959866 0.9584762 0.9111172 0.9340403
Class: 16   0.9889251   0.9965797      0.9223660      0.9995426 0.9223660 0.9889251 0.9544676
Class: 17   0.9633536   0.9982255      0.9567460      0.9985016 0.9567460 0.9633536 0.9599725
Class: 18   0.8485157   0.9978405      0.9473054      0.9930961 0.9473054 0.8485157 0.8951488
Class: 19   0.9654806   0.9992487      0.9826087      0.9984823 0.9826087 0.9654806 0.9739591
Class: 20   0.9773726   0.9985507      0.9650980      0.9990712 0.9650980 0.9773726 0.9711724
Class: 21   0.9750421   0.9992051      0.9784615      0.9990754 0.9784615 0.9750421 0.9767303
Class: 22   0.9768737   0.9979161      0.9500975      0.9990547 0.9500975 0.9768737 0.9632725
Class: 23   0.9652478   0.9990432      0.9736018      0.9987193 0.9736018 0.9652478 0.9693407
Class: 24   0.9507555   0.9984976      0.9653484      0.9978291 0.9653484 0.9507555 0.9579380
Class: 25   0.9828038   0.9991192      0.9794286      0.9992658 0.9794286 0.9828038 0.9811092
Class: 26   0.9807126   0.9988369      0.9649635      0.9993696 0.9649635 0.9807126 0.9727585
          Prevalence Detection Rate Detection Prevalence Balanced Accuracy
Class: 1  0.04017522     0.03972153           0.04036295         0.9940531
Class: 2  0.04444618     0.04047247           0.04247497         0.9542816
Class: 3  0.03629537     0.03565394           0.03832916         0.9897937
Class: 4  0.04454005     0.04076971           0.04278786         0.9566185
Class: 5  0.03753129     0.03499687           0.03731227         0.9651238
Class: 6  0.04339800     0.04022215           0.04200563         0.9625566
Class: 7  0.04198999     0.03806320           0.03997184         0.9522992
Class: 8  0.02897372     0.02598561           0.03121089         0.9458522
Class: 9  0.04000313     0.03911139           0.04161452         0.9876591
Class: 10 0.03662390     0.03493429           0.03668648         0.9761767
Class: 11 0.03815707     0.03634230           0.03934606         0.9747440
Class: 12 0.03000626     0.02989675           0.03254068         0.9968169
Class: 13 0.03326033     0.03255632           0.03371402         0.9888337
Class: 14 0.03272841     0.03125782           0.03293179         0.9767600
Class: 15 0.04321026     0.03936170           0.04106696         0.9546679
Class: 16 0.03947121     0.03903317           0.04231852         0.9927524
Class: 17 0.03915832     0.03771902           0.03942428         0.9807896
Class: 18 0.04375782     0.03712453           0.03918961         0.9231781
Class: 19 0.04211514     0.04066020           0.04137985         0.9823646
Class: 20 0.03939299     0.03850125           0.03989362         0.9879616
Class: 21 0.03571652     0.03482478           0.03559136         0.9871236
Class: 22 0.03903317     0.03812578           0.04012829         0.9873949
Class: 23 0.03527847     0.03404255           0.03496558         0.9821455
Class: 24 0.04217772     0.04009700           0.04153630         0.9746265
Class: 25 0.04092616     0.04022215           0.04106696         0.9909615
Class: 26 0.03163329     0.03102315           0.03214956         0.9897747
```

Let's see next the behavior of Bagging on testing dataset.

Bagging Test – SVM

The accuracy of bagging with SVM for testing data set is 0.9291 shows drop as without bagging the model accuracy is 0.9401.

```
> baggingModels(tstdf, "svm")
[1] "varianceEstimation"
                          [,1]
Mean of Accuracies     9.291e-01
Variance of Accuracies 2.837e-06
[1] "Mean"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
   0.92905109    0.92616873    0.92193153    0.93570956    0.04945255
[1] "Standard Deviation"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
  0.001684377   0.001753615   0.001758036   0.001608820   0.001621933
[1] "Variance"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
 2.837125e-06  3.075165e-06  3.090689e-06  2.588302e-06  2.630668e-06
          Sensitivity Specificity Pos Pred Value Neg Pred Value Precision    Recall        F1
Class: 1    0.9689179   0.9989727      0.9757848      0.9986684 0.9757848 0.9689179 0.9722920
Class: 2    0.8960373   0.9959417      0.8971154      0.9958649 0.8971154 0.8960373 0.8962258
Class: 3    0.9639024   0.9968499      0.9252252      0.9985165 0.9252252 0.9639024 0.9437985
Class: 4    0.8728498   0.9975826      0.9491935      0.9934251 0.9491935 0.8728498 0.9092440
Class: 5    0.8807976   0.9972151      0.9348214      0.9945967 0.9348214 0.8807976 0.9069204
Class: 6    0.9127850   0.9973442      0.9317073      0.9965118 0.9317073 0.9127850 0.9219168
Class: 7    0.9013311   0.9958386      0.9091667      0.9954198 0.9091667 0.9013311 0.9050510
Class: 8    0.8321733   0.9926307      0.7747126      0.9948360 0.7747126 0.8321733 0.8015682
Class: 9    0.9782121   0.9969263      0.9270270      0.9991251 0.9270270 0.9782121 0.9518372
Class: 10   0.9681672   0.9968409      0.9034483      0.9990200 0.9034483 0.9681672 0.9345964
Class: 11   0.9314741   0.9964742      0.9118483      0.9972670 0.9118483 0.9314741 0.9210529
Class: 12   0.9847605   0.9969571      0.9047059      0.9995480 0.9047059 0.9847605 0.9430089
Class: 13   0.9141627   0.9982192      0.9515464      0.9967083 0.9515464 0.9141627 0.9323666
Class: 14   0.9407710   0.9976599      0.9322404      0.9979611 0.9322404 0.9407710 0.9363760
Class: 15   0.8816761   0.9974491      0.9373832      0.9948728 0.9373832 0.8816761 0.9085497
Class: 16   0.9587263   0.9965421      0.9198238      0.9982867 0.9198238 0.9587263 0.9387498
Class: 17   0.9388033   0.9984744      0.9653680      0.9972185 0.9653680 0.9388033 0.9517842
Class: 18   0.8609585   0.9960961      0.9157025      0.9930126 0.9157025 0.8609585 0.8864830
Class: 19   0.9249926   0.9960475      0.8806818      0.9976244 0.8806818 0.9249926 0.9021307
Class: 20   0.9761631   0.9971190      0.9296296      0.9990502 0.9296296 0.9761631 0.9520583
Class: 21   0.9815564   0.9974755      0.9267760      0.9993959 0.9267760 0.9815564 0.9532719
Class: 22   0.9384731   0.9981712      0.9578947      0.9972201 0.9578947 0.9384731 0.9476849
Class: 23   0.9663193   0.9984461      0.9601942      0.9986727 0.9601942 0.9663193 0.9630270
Class: 24   0.9286056   0.9983179      0.9633333      0.9966031 0.9633333 0.9286056 0.9456093
Class: 25   0.9566833   0.9988573      0.9734513      0.9980967 0.9734513 0.9566833 0.9649330
Class: 26   0.9610808   0.9978456      0.9409326      0.9986003 0.9409326 0.9610808 0.9508014
          Prevalence Detection Rate Detection Prevalence Balanced Accuracy
Class: 1  0.04098540     0.03970803           0.04069343         0.9839453
Class: 2  0.03802920     0.03405109           0.03795620         0.9459895
Class: 3  0.03890511     0.03748175           0.04051095         0.9803762
Class: 4  0.04923358     0.04295620           0.04525547         0.9352162
Class: 5  0.04339416     0.03821168           0.04087591         0.9390064
Class: 6  0.03821168     0.03485401           0.03740876         0.9550646
Class: 7  0.04419708     0.03981752           0.04379562         0.9485849
Class: 8  0.02959854     0.02459854           0.03175182         0.9124020
Class: 9  0.03839416     0.03755474           0.04051095         0.9875692
Class: 10 0.02963504     0.02868613           0.03175182         0.9825040
Class: 11 0.03773723     0.03510949           0.03850365         0.9639741
Class: 12 0.02850365     0.02806569           0.03102190         0.9908588
Class: 13 0.03686131     0.03368613           0.03540146         0.9561910
Class: 14 0.03310219     0.03113139           0.03339416         0.9692155
Class: 15 0.04153285     0.03660584           0.03905109         0.9395626
Class: 16 0.03974453     0.03810219           0.04142336         0.9776342
Class: 17 0.04335766     0.04069343           0.04215328         0.9686389
Class: 18 0.04711679     0.04043796           0.04416058         0.9285273
Class: 19 0.03058394     0.02828467           0.03211679         0.9605200
Class: 20 0.03755474     0.03664234           0.03941606         0.9866411
Class: 21 0.03153285     0.03094891           0.03339416         0.9895160
Class: 22 0.04251825     0.03985401           0.04160584         0.9683222
Class: 23 0.03737226     0.03609489           0.03759124         0.9823827
Class: 24 0.04543796     0.04218978           0.04379562         0.9634617
Class: 25 0.04197080     0.04014599           0.04124088         0.9777703
Class: 26 0.03448905     0.03313869           0.03521898         0.9794632
```

Mohit Shankar Pursnani **Letter Recognition Dataset** mp5578@nyu.edu

Assignment - 3

Bagging Train – Tree

Let's do Bagging using Tree.

The accuracy of the model with Bagging is 0.4631 and variance is in the power of e-06. Bagging on Tree show increase in accuracy for training data set. Without Bagging the model accuracy was 0.39612.

```
> baggingModels(trdf, "tree")
[1] "varianceEstimation"
                          [,1]
Mean of Accuracies      0.4631000
Variance of Accuracies 0.0002133
[1] "Mean"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
    0.4630945     0.4410045     0.4544246     0.4717812     0.1375939
[1] "Standard Deviation"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
   0.01460564    0.01528082    0.01458278    0.01462184    0.02316193
[1] "Variance"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
 0.0002133246  0.0002335035  0.0002126574  0.0002137983  0.0005364749
```

	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1
Class: 1	0.9845454	0.9908060	0.7794961	0.9994783	0.779496124	0.9845454	0.8700469
Class: 2	0.3535385	0.9865931	0.7095396	0.9375361	0.709539595	0.3535385	0.4614747
Class: 3	0.7218665	0.9817177	0.5369388	0.9896779	0.536938776	0.7218665	0.6001802
Class: 4	NA	0.9838974	NA	NA	0.656142596	NA	NA
Class: 5	NA	0.9671631	NA	NA	0.129014675	NA	NA
Class: 6	NA	0.9598024	NA	NA	0.046089385	NA	NA
Class: 7	NA	0.9816846	NA	NA	0.573659491	NA	NA
Class: 8	NA	0.9689435	NA	NA	0.005614035	NA	NA
Class: 9	0.8231275	0.9890994	0.7480263	0.9916830	0.748026316	0.8231275	0.7759599
Class: 10	0.8716082	0.9847714	0.5944989	0.9964913	0.594498934	0.8716082	0.6945586
Class: 11	NA	0.9682003	NA	NA	0.209224652	NA	NA
Class: 12	0.5799972	0.9882974	0.6535577	0.9840734	0.653557692	0.5799972	0.6145429
Class: 13	0.9257051	0.9916070	0.7580046	0.9975148	0.758004640	0.9257051	0.8310969
Class: 14	NA	0.9842131	NA	NA	0.533372922	NA	NA
Class: 15	NA	0.9723541	NA	NA	0.345371429	NA	NA
Class: 16	0.4821809	0.9801647	0.5547874	0.9713510	0.554787431	0.4821809	0.5071171
Class: 17	NA	0.9762545	NA	NA	0.417738095	NA	NA
Class: 18	NA	0.9617051	NA	NA	0.024750499	NA	NA
Class: 19	NA	0.9640875	NA	NA	0.141285444	NA	NA
Class: 20	NA	0.9825020	NA	NA	0.578803922	NA	NA
Class: 21	NA	0.9845056	NA	NA	0.585362637	NA	NA
Class: 22	0.5534281	0.9845739	0.6328850	0.9776775	0.632884990	0.5534281	0.5843393
Class: 23	0.7979508	0.9911879	0.7561074	0.9922291	0.756107383	0.7979508	0.7690496
Class: 24	NA	0.9814467	NA	NA	0.598248588	NA	NA
Class: 25	NA	0.9653800	NA	NA	0.164000000	NA	NA
Class: 26	NA	0.9766991	NA	NA	0.284160584	NA	NA

	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Class: 1	0.031963392	0.0314627660	0.04036295	0.9876757
Class: 2	0.089948373	0.0301376721	0.04247497	0.6700658
Class: 3	0.030506884	0.0205804130	0.03832916	0.8517921
Class: 4	0.095977003	0.0280749374	0.04278786	NA
Class: 5	0.013319775	0.0048138298	0.03731227	NA
Class: 6	0.004595588	0.0019360138	0.04200563	NA
Class: 7	0.074105914	0.0229302253	0.03997184	NA
Class: 8	0.001319618	0.0001752190	0.03121089	NA
Class: 9	0.039099656	0.0311287547	0.04161452	0.9061135
Class: 10	0.025190081	0.0218100751	0.03668648	0.9281898
Class: 11	0.030926158	0.0082321652	0.03934606	NA
Class: 12	0.036675532	0.0212672090	0.03254068	0.7841473
Class: 13	0.027956821	0.0255553817	0.03371402	0.9586560
Class: 14	0.028333855	0.0175649249	0.03293179	NA
Class: 15	0.040242491	0.0141833542	0.04106696	NA
Class: 16	0.050914424	0.0234777847	0.04231852	0.7311728
Class: 17	0.044606539	0.0164690238	0.03942428	NA
Class: 18	0.003153942	0.0009699625	0.03918961	NA
Class: 19	0.011878911	0.0058463705	0.04137985	NA
Class: 20	0.046264862	0.0230905820	0.03989362	NA
Class: 21	0.049242804	0.0208338548	0.03559136	NA
Class: 22	0.046823373	0.0253965895	0.04012829	0.7690010
Class: 23	0.033936952	0.0264377347	0.03496558	0.8945693
Class: 24	0.111103723	0.0248490300	0.04153630	NA
Class: 25	0.012456977	0.0067349812	0.04106696	NA
Class: 26	0.019456352	0.0091356383	0.03214956	NA

Let's see next the behavior of Bagging on testing dataset.

Mohit Shankar Pursnani **Letter Recognition Dataset** mp5578@nyu.edu

Assignment - 3

Bagging Test – Tree

The accuracy of bagging with Tree for testing data set is 0.4779 shows increase as without bagging the model accuracy is 0.4567.

```
> baggingModels(tstdf, "tree")
[1] "varianceEstimation"
                          [,1]
Mean of Accuracies     0.4779000
Variance of Accuracies 0.0006025
[1] "Mean"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
    0.4778504     0.4560705     0.4645646     0.4911597     0.1254380
[1] "Standard Deviation"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
   0.02454566    0.02569679    0.02449835    0.02456684    0.02381126
[1] "Variance"
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
 0.0006024897  0.0006603248  0.0006001691  0.0006035294  0.0005669763
```

	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1
Class: 1	0.9797131	0.9921190	0.8128700	0.9992753	0.812869955	0.9797131	0.8884410
Class: 2	NA	0.9764570	NA	NA	0.397259615	NA	NA
Class: 3	0.6943241	0.9780531	0.4739640	0.9864169	0.473963964	0.6943241	0.5320565
Class: 4	0.4557123	0.9830910	0.6510484	0.9534920	0.651048387	0.4557123	0.5058472
Class: 5	NA	0.9637074	NA	NA	0.120669643	NA	NA
Class: 6	NA	0.9690245	NA	NA	0.182146341	NA	NA
Class: 7	NA	0.9749856	NA	NA	0.456625000	NA	NA
Class: 8	NA	0.9691494	NA	NA	0.031839080	NA	NA
Class: 9	0.6783495	0.9898060	0.7601351	0.9814492	0.760135135	0.6783495	0.7034109
Class: 10	NA	0.9841213	NA	NA	0.507528736	NA	NA
Class: 11	NA	0.9770470	NA	NA	0.431753555	NA	NA
Class: 12	0.6120288	0.9869644	0.5924118	0.9879661	0.592411765	0.6120288	0.6019741
Class: 13	NA	0.9882988	NA	NA	0.678917526	NA	NA
Class: 14	NA	0.9816963	NA	NA	0.463169399	NA	NA
Class: 15	NA	0.9750413	NA	NA	0.381028037	NA	NA
Class: 16	NA	0.9847634	NA	NA	0.650484581	NA	NA
Class: 17	NA	0.9820165	NA	NA	0.597012987	NA	NA
Class: 18	NA	0.9714931	NA	NA	0.375371901	NA	NA
Class: 19	NA	0.9679440	NA	NA	0.002045455	NA	NA
Class: 20	NA	0.9729351	NA	NA	0.323796296	NA	NA
Class: 21	NA	0.9824576	NA	NA	0.491584699	NA	NA
Class: 22	0.5755875	0.9886810	0.7422368	0.9730198	0.742236842	0.5755875	0.6346215
Class: 23	0.6570091	0.9868946	0.6661650	0.9812609	0.666165049	0.6570091	0.6408864
Class: 24	0.3251149	0.9796289	0.5772917	0.9219447	0.577291667	0.3251149	0.3725863
Class: 25	NA	0.9773494	NA	NA	0.465221239	NA	NA
Class: 26	NA	0.9779762	NA	NA	0.387772021	NA	NA

	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Class: 1	0.033773723	3.307847e-02	0.04069343	0.9859160
Class: 2	0.043386861	1.507847e-02	0.03795620	NA
Class: 3	0.032233577	1.920073e-02	0.04051095	0.8361886
Class: 4	0.073866788	2.946350e-02	0.04525547	0.7194017
Class: 5	0.012532847	4.932482e-03	0.04087591	NA
Class: 6	0.016877737	6.813869e-03	0.03740876	NA
Class: 7	0.056217153	1.999818e-02	0.04379562	NA
Class: 8	0.006085766	1.010949e-03	0.03175182	NA
Class: 9	0.048593066	3.079380e-02	0.04051095	0.8340777
Class: 10	0.019906934	1.611496e-02	0.03175182	NA
Class: 11	0.057344891	1.662409e-02	0.03850365	NA
Class: 12	0.030038321	1.837774e-02	0.03102190	0.7994966
Class: 13	0.030136861	2.403467e-02	0.03540146	NA
Class: 14	0.026000000	1.546715e-02	0.03339416	NA
Class: 15	0.038034672	1.487956e-02	0.03905109	NA
Class: 16	0.056647810	2.694526e-02	0.04142336	NA
Class: 17	0.060565693	2.516606e-02	0.04215328	NA
Class: 18	0.041684307	1.657664e-02	0.04416058	NA
Class: 19	0.000189781	6.569343e-05	0.03211679	NA
Class: 20	0.023065693	1.276277e-02	0.03941606	NA
Class: 21	0.037239051	1.641606e-02	0.03339416	NA
Class: 22	0.056739051	3.088139e-02	0.04160584	0.7821343
Class: 23	0.043076642	2.504197e-02	0.03759124	0.8219518
Class: 24	0.099919708	2.528285e-02	0.04379562	0.6523719
Class: 25	0.029828467	1.918613e-02	0.04124088	NA
Class: 26	0.026014599	1.365693e-02	0.03521898	NA

KNN Bagging

Let's now do Bagging with KNN and see the behavior.

The accuracy of Bagging with KNN model is 0.9535 and without bagging the accuracy is 0.95419. The accuracy is almost same, the variance with Bagging reduced further, it is in the power of e-08, while without bagging the variance is e-06.

```
> ###############################################################################
> #                          KNN Bagging                                        #
> ###############################################################################
> trdf <- trainingDataset_withoutOutliers
> tstdf <- testingDataset_withoutOutliers
>
> label <- trdf$Letter
> trdf_knn = trdf[-which(names(trdf)=="Letter")]
> tstdf_knn = tstdf[-which(names(tstdf)=="Letter")]
> trclass_knn=factor(trdf[which(names(trdf)=="Letter")])
Warning message:
In xtfrm.data.frame(x) : cannot xtfrm data frames
> tstclass_knn=factor(tstdf[which(names(tstdf)=="Letter")])
Warning message:
In xtfrm.data.frame(x) : cannot xtfrm data frames
> runModel<-function(df1, df2) {
+   knn(df1[-which(names(df1)=="Letter")],df2[-which(names(df2)=="Letter")],label, k = 1)
+ }
>
> lapplyrunmodel <- function(x) runModel(trdf, tstdf)
> models<-lapply(1:5,lapplyrunmodel)
> bagging_cfm <- lapply(models, FUN=function(P) {
+   pred_cfm<-caret::confusionMatrix(table(P,tstdf$Letter))
+   pred_cfm
+ })
> bagging.perf<-as.data.frame(do.call('rbind',lapply(bagging_cfm,FUN=function(cfm)c(cfm$overall))))
>
> acc_varEstp <- bagging.perf$Accuracy
> mean_varEstp = signif(mean(acc_varEstp),4)
> var_varEstp = signif(var(acc_varEstp),4)
> varEstp = data.frame(mean_varEstp,var_varEstp)
> names(varEstp) = c("Mean of Accuracies","Variance of Accuracies")
> varianceEstimation <- t(varEstp)
> print("varianceEstimation")
[1] "varianceEstimation"
> print(varianceEstimation)
                          [,1]
Mean of Accuracies     9.535e-01
Variance of Accuracies 9.324e-08
> bagging.perf.mean <-apply(bagging.perf[bagging.perf$AccuracyPValue<0.01,-c(6:7)],2,mean)
> print("Mean")
[1] "Mean"
> print(bagging.perf.mean)
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
   0.95354015    0.95165939    0.94762981    0.95896122    0.04525547
> bagging.perf.std<-apply(bagging.perf[bagging.perf$AccuracyPValue<0.01,-c(6:7)],2,sd)
> print("Standard Deviation")
[1] "Standard Deviation"
> print(bagging.perf.std)
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
 0.0003053504  0.0003175966  0.0003225945  0.0002877448  0.0000000000
> bagging.perf.var<-apply(bagging.perf[bagging.perf$AccuracyPValue<0.01,-c(6:7)],2,sd)
> print("Variance")
[1] "Variance"
> print(bagging.perf.var)
     Accuracy         Kappa AccuracyLower AccuracyUpper  AccuracyNull
 0.0003053504  0.0003175966  0.0003225945  0.0002877448  0.0000000000
> for(c in bagging_cfm) {
+   confusionMatrixByClass <- c$byClass
+   confusion_matrix_sd <- c$byClass
+   break
+ }
> y <- 0
> z <- c()
> cfm_len <- length(bagging_cfm)
> for(column in colnames(confusionMatrixByClass)) {
+   for(row in rownames(confusionMatrixByClass)) {
+     for (x in bagging_cfm) {
+       my_mat <- as.matrix(x$byClass)
+       y <- y + my_mat[row,column]
+       z <- c(z,my_mat[row,column])
+     }
+     confusionMatrixByClass[row,column] <- y/cfm_len
+     confusion_matrix_sd[row, column] <- sd(z)
+     y <- 0
+     z <- c()
+   }
+ }
> confusionMatrixByClass
```

	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1
Class: 1	0.9955157	0.9996196	0.9910714	0.9998097	0.9910714	0.9955157	0.9932886
Class: 2	0.9326923	0.9954476	0.8899306	0.9973394	0.8899306	0.9326923	0.9108041
Class: 3	0.9603604	0.9988589	0.9726276	0.9983273	0.9726276	0.9603604	0.9664543
Class: 4	0.9346774	0.9969801	0.9361854	0.9969039	0.9361854	0.9346774	0.9354284
Class: 5	0.9455357	0.9981355	0.9557891	0.9976799	0.9557891	0.9455357	0.9506259
Class: 6	0.9512195	0.9963223	0.9095513	0.9981009	0.9095513	0.9512195	0.9299096
Class: 7	0.9550000	0.9982061	0.9606062	0.9979395	0.9606062	0.9550000	0.9577939
Class: 8	0.8954023	0.9941576	0.8341524	0.9965617	0.8341524	0.8954023	0.8636493
Class: 9	0.9720721	0.9990491	0.9773550	0.9988211	0.9773550	0.9720721	0.9747056
Class: 10	0.9597701	0.9986430	0.9586732	0.9986807	0.9586732	0.9597701	0.9592201
Class: 11	0.8919431	0.9969634	0.9217022	0.9956784	0.9217022	0.8919431	0.9065625
Class: 12	0.9705882	0.9996610	0.9892285	0.9990589	0.9892285	0.9705882	0.9798148
Class: 13	0.9577320	0.9994325	0.9841099	0.9984503	0.9841099	0.9577320	0.9707380
Class: 14	0.9650273	0.9991316	0.9746398	0.9987922	0.9746398	0.9650273	0.9698026
Class: 15	0.9588785	0.9973414	0.9361855	0.9983273	0.9361855	0.9588785	0.9473816
Class: 16	0.8986784	0.9982867	0.9577457	0.9956332	0.9577457	0.8986784	0.9272683
Class: 17	0.9826840	0.9980568	0.9570078	0.9992370	0.9570078	0.9826840	0.9696736
Class: 18	0.9363636	0.9961436	0.9181572	0.9970574	0.9181572	0.9363636	0.9271553
Class: 19	0.9625000	0.9992459	0.9769318	0.9987563	0.9769318	0.9625000	0.9696603
Class: 20	0.9546296	0.9988602	0.9717291	0.9981396	0.9717291	0.9546296	0.9631026
Class: 21	0.9639344	0.9994336	0.9832775	0.9987549	0.9832775	0.9639344	0.9735081
Class: 22	0.9482456	0.9990099	0.9765193	0.9977561	0.9765193	0.9482456	0.9621717
Class: 23	0.9883495	0.9990140	0.9751010	0.9995447	0.9751010	0.9883495	0.9816774
Class: 24	0.9516667	0.9980916	0.9580672	0.9977869	0.9580672	0.9516667	0.9548536
Class: 25	0.9849558	0.9985915	0.9678296	0.9993524	0.9678296	0.9849558	0.9763157
Class: 26	0.9772021	0.9989786	0.9721750	0.9991676	0.9721750	0.9772021	0.9746769

	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Class: 1	0.04069343	0.04051095	0.04087591	0.9975676
Class: 2	0.03795620	0.03540146	0.03978102	0.9640700
Class: 3	0.04051095	0.03890511	0.04000000	0.9796096
Class: 4	0.04525547	0.04229927	0.04518248	0.9658288
Class: 5	0.04087591	0.03864964	0.04043796	0.9718356
Class: 6	0.03740876	0.03558394	0.03912409	0.9737709
Class: 7	0.04379562	0.04182482	0.04354015	0.9766031
Class: 8	0.03175182	0.02843066	0.03408759	0.9447799
Class: 9	0.04051095	0.03937956	0.04029197	0.9855606
Class: 10	0.03175182	0.03047445	0.03178832	0.9792066
Class: 11	0.03850365	0.03434307	0.03726277	0.9444532
Class: 12	0.03102190	0.03010949	0.03043796	0.9851246
Class: 13	0.03540146	0.03390511	0.03445255	0.9785822
Class: 14	0.03339416	0.03222628	0.03306569	0.9820795
Class: 15	0.03905109	0.03744526	0.04000000	0.9781100
Class: 16	0.04142336	0.03722628	0.03886861	0.9484826
Class: 17	0.04215328	0.04142336	0.04328467	0.9903704
Class: 18	0.04416058	0.04135036	0.04503650	0.9662536
Class: 19	0.03211679	0.03091241	0.03164234	0.9808729
Class: 20	0.03941606	0.03762774	0.03872263	0.9767449
Class: 21	0.03339416	0.03218978	0.03273723	0.9816840
Class: 22	0.04160584	0.03945255	0.04040146	0.9736278
Class: 23	0.03759124	0.03715328	0.03810219	0.9936818
Class: 24	0.04379562	0.04167883	0.04350365	0.9748791
Class: 25	0.04124088	0.04062044	0.04197080	0.9917737
Class: 26	0.03521898	0.03441606	0.03540146	0.9880903

Random Forest

Random forest is an ensemble learning method for classification that operates by constructing a multitude of decision trees at training time. For classification, the output of the random forest is the class selected by most trees.

Let's look now by perfroming randomForest on the dataset. The default value of number of trees (nTrees) is 500 for this function.

```
> ################################################################################
> #                          Random Forest                                       #
> ################################################################################
>
> trdf <- trainingDataset_withoutOutliers
> tstdf <- testingDataset_withoutOutliers
>
> rf_model<-randomForest(Letter~.,data=trdf)
> rf_pred<-predict(rf_model,tstdf[,-c(which(names(trdf)=="Letter"))])
> rf_mtab<-table(tstdf$Letter,rf_pred)
> rf_cmx<-caret::confusionMatrix(rf_mtab)
> rf_cmx
Confusion Matrix and Statistics

   rf_pred
      1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24
  1 222   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
  2   0 197   0   1   0   0   1   1   0   0   0   0   0   0   0   0   0   5   0   0   1   1   0   1
  3   0   0 210   0   2   1   5   0   0   0   0   0   0   0   2   0   0   0   0   0   2   0   0   0
  4   0   2   0 238   0   0   0   2   0   0   0   0   0   2   4   0   0   0   0   0   0   0   0   0
  5   0   0   0   0 212   0   7   0   0   0   2   0   0   0   0   0   0   0   0   1   0   0   0   1
  6   0   1   0   0   0 194   0   0   0   1   0   0   0   0   0   6   1   0   0   2   0   0   0   0
  7   0   0   1   2   1   0 233   0   0   0   0   0   0   0   0   1   0   1   1   0   0   0   0   0
  8   0   0   0   1   0   0   1 162   0   0   4   0   0   0   2   0   0   4   0   0   0   0   0   0
  9   0   0   0   1   0   0   0   0 211   8   0   0   0   0   0   1   0   0   0   1   0   0   0   0
  10  0   0   0   0   1   1   0   0   7 160   1   1   0   0   2   0   1   0   0   0   0   0   0   0
  11  0   1   0   0   1   0   0   2   0   0 198   0   1   0   0   0   0   4   0   0   1   0   0   3
  12  0   0   0   0   2   0   2   0   0   0   0 163   0   0   0   0   0   2   0   0   0   0   0   1
  13  0   2   0   0   0   0   0   0   0   0   1   0 186   0   0   0   1   0   0   0   0   2   2   0
  14  0   1   0   0   0   0   0   1   0   0   1   0   0 176   2   0   0   1   0   0   0   0   1   0
  15  0   1   1   1   0   0   0   0   0   0   0   0   0   0 207   0   3   1   0   0   0   0   0   0
  16  0   0   0   0   0   6   1   0   0   0   0   0   0   0   1 218   0   0   0   0   0   0   1   0
  17  0   0   0   0   0   0   1   0   0   0   0   0   0   0   1   0 227   1   0   0   0   0   0   0
  18  0   4   0   2   0   1   0   0   0   0   0   0   0   2   0   0   0 232   0   0   0   0   0   1
  19  0   2   0   0   1   1   0   2   0   0   0   0   0   0   0   0   1   1 168   0   0   0   0   0
  20  1   1   1   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   2 208   0   0   0   0
  21  0   0   0   0   0   0   0   3   0   0   0   0   3   0   0   0   1   0   0   0 176   0   0   0
  22  0   7   0   0   0   0   0   0   0   0   0   0   0   2   1   1   0   0   0   0   0 214   2   0
  23  0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 205   0
  24  0   0   0   0   1   0   0   0   0   0   2   0   0   0   0   0   1   1   0   0   0   0   0 235
  25  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   2   0   0   0   0
  26  0   2   0   0   0   2   0   0   0   2   0   0   0   0   0   0   3   0   0   0   0   0   0   0
   rf_pred
     25  26
  1   0   0
  2   0   0
  3   0   0
  4   0   0
  5   0   1
  6   0   0
  7   0   0
  8   0   0
  9   0   0
  10  0   0
  11  0   0
  12  0   0
  13  0   0
  14  0   0
  15  0   0
  16  0   0
  17  0   1
  18  0   0
  19  0   0
  20  2   0
  21  0   0
  22  1   0
  23  0   0
  24  0   0
  25 223  0
  26  0 184
```

```
Overall Statistics

               Accuracy : 0.9597
                 95% CI : (0.9541, 0.9647)
    No Information Rate : 0.0462
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.958

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
Sensitivity           0.99552  0.89140  0.98592  0.96748  0.95928  0.94175  0.92460  0.93642
Specificity           0.99981  0.99791  0.99772  0.99809  0.99772  0.99791  0.99866  0.99774
Pos Pred Value        0.99552  0.94712  0.94595  0.95968  0.94643  0.94634  0.97083  0.93103
Neg Pred Value        0.99981  0.99545  0.99943  0.99847  0.99829  0.99773  0.99637  0.99793
Prevalence            0.04069  0.04033  0.03887  0.04489  0.04033  0.03759  0.04599  0.03157
Detection Rate        0.04051  0.03595  0.03832  0.04343  0.03869  0.03540  0.04252  0.02956
Detection Prevalence  0.04069  0.03796  0.04051  0.04526  0.04088  0.03741  0.04380  0.03175
Balanced Accuracy     0.99766  0.94466  0.99182  0.98278  0.97850  0.96983  0.96163  0.96708
                     Class: 9 Class: 10 Class: 11 Class: 12 Class: 13 Class: 14 Class: 15 Class: 16
Sensitivity           0.96789   0.93567   0.93839   0.99390   0.97895   0.96703   0.93243   0.96035
Specificity           0.99791   0.99736   0.99753   0.99868   0.99849   0.99868   0.99867   0.99829
Pos Pred Value        0.95045   0.91954   0.93839   0.95882   0.95876   0.96175   0.96729   0.96035
Neg Pred Value        0.99867   0.99793   0.99753   0.99981   0.99924   0.99887   0.99715   0.99829
Prevalence            0.03978   0.03120   0.03850   0.02993   0.03467   0.03321   0.04051   0.04142
Detection Rate        0.03850   0.02920   0.03613   0.02974   0.03394   0.03212   0.03777   0.03978
Detection Prevalence  0.04051   0.03175   0.03850   0.03102   0.03540   0.03339   0.03905   0.04142
Balanced Accuracy     0.98290   0.96652   0.96796   0.99629   0.98872   0.98286   0.96555   0.97932
                     Class: 17 Class: 18 Class: 19 Class: 20 Class: 21 Class: 22 Class: 23 Class: 24
Sensitivity            0.94979   0.91700   0.98246   0.97653   0.96703   0.98618   0.97156   0.97107
Specificity            0.99924   0.99809   0.99849   0.99848   0.99868   0.99734   0.99981   0.99905
Pos Pred Value         0.98268   0.95868   0.95455   0.96296   0.96175   0.93860   0.99515   0.97917
Neg Pred Value         0.99771   0.99599   0.99943   0.99905   0.99887   0.99943   0.99886   0.99866
Prevalence             0.04361   0.04617   0.03120   0.03887   0.03321   0.03960   0.03850   0.04416
Detection Rate         0.04142   0.04234   0.03066   0.03796   0.03212   0.03905   0.03741   0.04288
Detection Prevalence   0.04215   0.04416   0.03212   0.03942   0.03339   0.04161   0.03759   0.04380
Balanced Accuracy      0.97451   0.95754   0.99047   0.98750   0.98286   0.99176   0.98569   0.98506
                     Class: 25 Class: 26
Sensitivity            0.98673   0.98925
Specificity            0.99943   0.99830
Pos Pred Value         0.98673   0.95337
Neg Pred Value         0.99943   0.99962
Prevalence             0.04124   0.03394
Detection Rate         0.04069   0.03358
Detection Prevalence   0.04124   0.03522
Balanced Accuracy      0.99308   0.99377
```

Accuracy

```
>
> (rf_accuracy<-sum(diag(rf_mtab))/sum(rf_mtab))
[1] 0.9596715
>
> rf_cmx$overall
      Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull AccuracyPValue
    0.95967153     0.95803676     0.95412118     0.96472529     0.04616788     0.00000000
 McnemarPValue
           NaN
```

Random Forest – 50 Trees

```
> ################################################################################
> #                        Random Forest 50 Trees                                #
> ################################################################################
>
> rf50_model<-randomForest(Letter~.,data=trdf,ntree=50)
> rf50_pred<-predict(rf50_model,tstdf[,-c(which(names(trdf)=="Letter"))])
> rf50_mtab<-table(tstdf$Letter,rf50_pred)
> rf50_cmx<-caret::confusionMatrix(rf50_mtab)
> rf50_mtab
```

Mohit Shankar Pursnani **Letter Recognition Dataset** mp5578@nyu.edu

Assignment - 3

```
> rf50_mtab
    rf50_pred
       1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24
1  219   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   1   1   0   1   0   0   0
2    0 197   0   1   0   0   1   2   0   0   0   0   0   0   0   0   0   4   1   0   0   1   0   1
3    0   0 210   0   2   2   3   0   0   0   0   1   0   0   2   0   0   0   0   0   2   0   0   0
4    0   3   0 236   0   0   0   3   0   0   0   0   0   1   5   0   0   0   0   0   0   0   0   0
5    0   0   0   0 209   0   6   0   0   0   4   0   0   0   0   0   2   0   1   0   0   0   0   1
6    0   2   0   0   0 196   0   0   0   0   0   0   0   0   0   5   0   0   0   2   0   0   0   0
7    0   0   2   1   2   0 229   0   0   0   1   0   0   0   1   1   0   2   1   0   0   0   0   0
8    0   2   0   1   0   0   1 162   0   0   4   0   0   0   1   0   0   3   0   0   0   0   0   0
9    0   0   0   1   0   0   0   0 214   6   0   0   0   0   0   0   0   0   0   1   0   0   0   0
10   0   0   0   0   1   0   0   1   5 163   1   1   0   0   1   0   0   1   0   0   0   0   0   0
11   0   1   0   0   1   0   0   3   0   0 199   0   0   0   0   0   0   4   0   0   1   0   0   2
12   0   0   0   0   2   0   2   0   0   0   1 161   0   0   0   0   1   1   1   0   0   0   0   1
13   0   2   0   0   0   0   1   0   0   0   2   0 182   0   0   1   1   1   0   0   0   2   2   0
14   0   2   1   3   0   0   0   4   0   0   0   0   1 170   0   0   0   1   0   0   1   0   0   0
15   0   1   1   4   0   0   0   0   0   0   0   0   0   0 202   0   5   1   0   0   0   0   0   0
16   0   0   0   0   0   9   1   0   0   0   0   0   0   0   1 214   0   1   0   0   0   0   1   0
17   0   0   0   0   0   0   1   0   0   0   0   0   0   0   2   1 225   1   0   0   0   0   0   0
18   0   5   0   1   1   0   0   2   0   0   0   0   0   2   0   0   0 230   0   0   0   0   0   1
19   0   0   1   0   0   1   1   2   0   0   0   0   0   0   0   0   1   2 166   0   0   0   0   1
20   0   0   2   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   3 208   0   0   0   0
21   0   0   0   0   0   0   1   3   0   0   0   0   2   1   0   0   1   0   0   0 174   0   0   0
22   1   4   0   0   0   0   3   0   0   0   0   0   0   2   0   2   0   0   0   0   0 212   2   0
23   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 205   0
24   0   0   0   0   1   1   0   0   0   2   1   0   0   0   0   0   1   0   0   0   0   0   0 234
25   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   2   0   0   0
26   0   1   0   1   0   2   0   0   0   1   0   0   0   0   0   0   3   1   0   0   0   0   0   0
   rf50_pred
     25  26
1     0   0
2     0   0
3     0   0
4     0   0
5     0   1
6     0   0
7     0   0
8     0   0
9     0   0
10    0   0
11    0   0
12    0   0
13    0   0
14    0   0
15    0   0
16    0   0
17    0   1
18    0   0
19    0   1
20    2   0
21    1   0
22    2   0
23    0   0
24    0   0
25  221   0
26    0 184
```

Accuracy

```
> (rf50_accuracy<-sum(diag(rf50_mtab))/sum(rf50_mtab))
[1] 0.9529197
```

Overall Statistics

```
> rf50_cmx$overall
      Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull AccuracyPValue
    0.95291971     0.95101165     0.94697449     0.95837640     0.04635036     0.00000000
 McnemarPValue
           NaN
```

Statistics by Class

```
> rf50_cmx$byClass
         Sensitivity Specificity Pos Pred Value Neg Pred Value Precision    Recall        F1
Class: 1   0.9954545   0.9992395      0.9820628      0.9998098 0.9820628 0.9954545 0.9887133
Class: 2   0.8914027   0.9979083      0.9471154      0.9954476 0.9471154 0.8914027 0.9184149
Class: 3   0.9677419   0.9977199      0.9459459      0.9986687 0.9459459 0.9677419 0.9567198
Class: 4   0.9477912   0.9977060      0.9516129      0.9975153 0.9516129 0.9477912 0.9496982
Class: 5   0.9543379   0.9971488      0.9330357      0.9980974 0.9330357 0.9543379 0.9435666
Class: 6   0.9289100   0.9982919      0.9560976      0.9971564 0.9560976 0.9289100 0.9423077
Class: 7   0.9123506   0.9978963      0.9541667      0.9958015 0.9541667 0.9123506 0.9327902
Class: 8   0.8901099   0.9977350      0.9310345      0.9962307 0.9310345 0.8901099 0.9101124
Class: 9   0.9771689   0.9984794      0.9639640      0.9990491 0.9639640 0.9771689 0.9705215
Class: 10  0.9476744   0.9979277      0.9367816      0.9983038 0.9367816 0.9476744 0.9421965
Class: 11  0.9255814   0.9977208      0.9431280      0.9969634 0.9431280 0.9255814 0.9342723
Class: 12  0.9877301   0.9983073      0.9470588      0.9996234 0.9470588 0.9877301 0.9669670
Class: 13  0.9837838   0.9977337      0.9381443      0.9994325 0.9381443 0.9837838 0.9604222
Class: 14  0.9659091   0.9975490      0.9289617      0.9988673 0.9289617 0.9659091 0.9470752
Class: 15  0.9395349   0.9977208      0.9439252      0.9975313 0.9439252 0.9395349 0.9417249
Class: 16  0.9553571   0.9975266      0.9427313      0.9980963 0.9427313 0.9553571 0.9490022
Class: 17  0.9375000   0.9988550      0.9740260      0.9971423 0.9740260 0.9375000 0.9554140
Class: 18  0.9055118   0.9977038      0.9504132      0.9954181 0.9504132 0.9055118 0.9274194
Class: 19  0.9485714   0.9981150      0.9431818      0.9983032 0.9431818 0.9485714 0.9458689
Class: 20  0.9811321   0.9984814      0.9629630      0.9992401 0.9629630 0.9811321 0.9719626
Class: 21  0.9613260   0.9983016      0.9508197      0.9986785 0.9508197 0.9613260 0.9560440
Class: 22  0.9860465   0.9969611      0.9298246      0.9994288 0.9298246 0.9860465 0.9571106
Class: 23  0.9761905   0.9998102      0.9951456      0.9990520 0.9951456 0.9761905 0.9855769
Class: 24  0.9709544   0.9988547      0.9750000      0.9986641 0.9750000 0.9709544 0.9729730
Class: 25  0.9778761   0.9990483      0.9778761      0.9990483 0.9778761 0.9778761 0.9778761
Class: 26  0.9839572   0.9982996      0.9533679      0.9994326 0.9533679 0.9839572 0.9684211
          Prevalence Detection Rate Detection Prevalence Balanced Accuracy
Class: 1  0.04014599     0.03996350           0.04069343         0.9973470
Class: 2  0.04032847     0.03594891           0.03795620         0.9446555
Class: 3  0.03959854     0.03832117           0.04051095         0.9827309
Class: 4  0.04543796     0.04306569           0.04525547         0.9727486
Class: 5  0.03996350     0.03813869           0.04087591         0.9757434
Class: 6  0.03850365     0.03576642           0.03740876         0.9636009
Class: 7  0.04580292     0.04178832           0.04379562         0.9551235
Class: 8  0.03321168     0.02956204           0.03175182         0.9439224
Class: 9  0.03996350     0.03905109           0.04051095         0.9878242
Class: 10 0.03138686     0.02974453           0.03175182         0.9728010
Class: 11 0.03923358     0.03631387           0.03850365         0.9616511
Class: 12 0.02974453     0.02937956           0.03102190         0.9930187
Class: 13 0.03375912     0.03321168           0.03540146         0.9907587
Class: 14 0.03211679     0.03102190           0.03339416         0.9817291
Class: 15 0.03923358     0.03686131           0.03905109         0.9686278
Class: 16 0.04087591     0.03905109           0.04142336         0.9764419
Class: 17 0.04379562     0.04105839           0.04215328         0.9681775
Class: 18 0.04635036     0.04197080           0.04416058         0.9516078
Class: 19 0.03193431     0.03029197           0.03211679         0.9733432
Class: 20 0.03868613     0.03795620           0.03941606         0.9898067
Class: 21 0.03302920     0.03175182           0.03339416         0.9798138
Class: 22 0.03923358     0.03868613           0.04160584         0.9915038
Class: 23 0.03832117     0.03740876           0.03759124         0.9880004
Class: 24 0.04397810     0.04270073           0.04379562         0.9849046
Class: 25 0.04124088     0.04032847           0.04124088         0.9884622
Class: 26 0.03412409     0.03357664           0.03521898         0.9911284
```

Random Forest – 100 Trees

```
> ################################################################################
> #                        Random Forest 100 Trees                               #
> ################################################################################
> # Number of trees - 100
> rf100_model<-randomForest(Letter~.,data=trdf,ntree=100)
> rf100_pred<-predict(rf100_model,tstdf[,-c(which(names(trdf)=="Letter"))])
> rf100_mtab<-table(tstdf$Letter,rf100_pred)
> rf100_cmx<-caret::confusionMatrix(rf100_mtab)
> rf100_mtab
```

Mohit Shankar Pursnani **Letter Recognition Dataset** mp5578@nyu.edu

Assignment - 3

```
   rf100_pred
      1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24
1   221   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   0   0   0   0
2     0 201   0   1   0   0   0   2   0   0   0   0   0   0   0   0   0   3   0   0   0   0   0   1
3     0   0 210   0   3   1   5   0   0   0   0   0   0   0   1   0   0   0   0   0   2   0   0   0
4     0   2   0 238   0   0   0   2   0   0   0   0   0   2   4   0   0   0   0   0   0   0   0   0
5     0   0   0   0 211   0   8   0   0   0   3   0   0   0   0   0   0   0   0   1   0   0   0   1
6     0   2   0   0   0 195   0   0   0   1   0   0   0   0   0   5   0   0   0   2   0   0   0   0
7     0   0   1   2   1   0 232   0   0   0   0   0   0   0   1   1   0   2   0   0   0   0   0   0
8     0   1   0   3   0   0   0 164   0   0   2   0   0   0   1   0   0   3   0   0   0   0   0   0
9     0   0   0   0   0   1   0   0 212   6   0   0   0   0   0   0   0   0   2   1   0   0   0   0
10    0   0   0   0   1   1   0   0   5 162   1   1   0   0   1   1   1   0   0   0   0   0   0   0
11    0   1   0   0   1   0   1   3   0   0 198   0   0   0   0   0   0   4   0   0   1   0   0   2
12    0   0   0   0   2   0   1   0   0   0   0 163   0   0   0   0   2   2   0   0   0   0   0   0
13    0   2   0   0   0   0   1   0   0   0   2   0 179   1   1   1   1   0   0   0   2   3   1   0
14    0   2   0   1   0   0   0   3   0   0   1   0   2 172   1   0   0   0   0   0   0   0   1   0
15    0   1   1   2   0   0   0   0   0   0   0   0   0   0 205   0   5   0   0   0   0   0   0   0
16    0   0   0   1   0   9   1   1   0   0   0   0   0   0   0 214   0   0   0   0   0   1   0   0
17    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 229   1   0   0   0   0   0   0
18    0   4   0   1   0   0   0   0   0   0   0   0   0   3   0   1   0 232   0   0   0   0   0   1
19    0   0   0   0   2   1   0   3   0   0   0   0   0   0   0   0   1   1 167   0   0   0   0   0
20    1   1   1   0   0   1   0   1   0   0   1   0   0   0   0   0   0   0   3 205   0   0   0   0
21    0   0   0   0   0   0   0   2   0   0   0   0   2   0   1   0   1   0   0   0 177   0   0   0
22    0   6   0   0   0   0   2   0   0   0   0   0   0   2   0   1   0   1   0   0   0 213   2   0
23    0   0   0   0   0   0   1   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0 204   0
24    0   0   0   0   1   0   0   1   0   1   1   0   0   0   0   0   0   0   0   0   0   0   0 236
25    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   2   0   0   0   0
26    0   2   0   1   0   0   0   0   0   3   0   0   0   0   0   0   4   0   0   0   0   0   0   0
   rf100_pred
     25  26
1     0   0
2     0   0
3     0   0
4     0   0
5     0   0
6     0   0
7     0   0
8     0   0
9     0   0
10    0   0
11    0   0
12    0   0
13    0   0
14    0   0
15    0   0
16    0   0
17    0   1
18    0   0
19    0   1
20    2   0
21    0   0
22    1   0
23    0   0
24    0   0
25  223   0
26    0 183
```

Overall Statistics:

```
> rf100_cmx$overall
      Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull AccuracyPValue
     0.9572993      0.9555689      0.9516061      0.9624987      0.0459854      0.0000000
 McnemarPValue
           NaN
```

Accuracy:

```
> (rf100_accuracy<-sum(diag(rf100_test_mtab))/sum(rf100_test_mtab))
[1] 0.9562044
```

Statistics by Class:

```
> rf100_cmx$byClass
```

	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1
Class: 1	0.9954955	0.9996196	0.9910314	0.9998098	0.9910314	0.9954955	0.9932584
Class: 2	0.8933333	0.9986679	0.9663462	0.9954476	0.9663462	0.8933333	0.9284065
Class: 3	0.9859155	0.9977217	0.9459459	0.9994294	0.9459459	0.9859155	0.9655172
Class: 4	0.9520000	0.9980880	0.9596774	0.9977064	0.9596774	0.9520000	0.9558233
Class: 5	0.9504505	0.9975276	0.9419643	0.9979072	0.9419643	0.9504505	0.9461883
Class: 6	0.9330144	0.9981028	0.9512195	0.9973460	0.9512195	0.9330144	0.9420290
Class: 7	0.9206349	0.9984698	0.9666667	0.9961832	0.9666667	0.9206349	0.9430894
Class: 8	0.9010989	0.9981125	0.9425287	0.9966076	0.9425287	0.9010989	0.9213483
Class: 9	0.9769585	0.9980999	0.9549550	0.9990491	0.9549550	0.9769585	0.9658314
Class: 10	0.9364162	0.9977388	0.9310345	0.9979269	0.9310345	0.9364162	0.9337176
Class: 11	0.9473684	0.9975337	0.9383886	0.9979123	0.9383886	0.9473684	0.9428571
Class: 12	0.9939024	0.9986832	0.9588235	0.9998117	0.9588235	0.9939024	0.9760479
Class: 13	0.9781421	0.9971682	0.9226804	0.9992433	0.9226804	0.9781421	0.9496021
Class: 14	0.9555556	0.9979245	0.9398907	0.9984897	0.9398907	0.9555556	0.9476584
Class: 15	0.9447005	0.9982899	0.9579439	0.9977212	0.9579439	0.9447005	0.9512761
Class: 16	0.9553571	0.9975266	0.9427313	0.9980963	0.9427313	0.9553571	0.9490022
Class: 17	0.9385246	0.9996180	0.9913420	0.9971423	0.9913420	0.9385246	0.9642105
Class: 18	0.9317269	0.9980883	0.9586777	0.9967545	0.9586777	0.9317269	0.9450102
Class: 19	0.9653179	0.9983041	0.9488636	0.9988688	0.9488636	0.9653179	0.9570201
Class: 20	0.9715640	0.9979123	0.9490741	0.9988602	0.9490741	0.9715640	0.9601874
Class: 21	0.9619565	0.9988671	0.9672131	0.9986785	0.9672131	0.9619565	0.9645777
Class: 22	0.9815668	0.9971499	0.9342105	0.9992384	0.9342105	0.9815668	0.9573034
Class: 23	0.9807692	0.9996206	0.9902913	0.9992416	0.9902913	0.9807692	0.9855072
Class: 24	0.9792531	0.9992365	0.9833333	0.9990458	0.9833333	0.9792531	0.9812890
Class: 25	0.9867257	0.9994290	0.9867257	0.9994290	0.9867257	0.9867257	0.9867257
Class: 26	0.9891892	0.9981114	0.9481865	0.9996217	0.9481865	0.9891892	0.9682540

	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Class: 1	0.04051095	0.04032847	0.04069343	0.9975576
Class: 2	0.04105839	0.03667883	0.03795620	0.9460006
Class: 3	0.03886861	0.03832117	0.04051095	0.9918186
Class: 4	0.04562044	0.04343066	0.04525547	0.9750440
Class: 5	0.04051095	0.03850365	0.04087591	0.9739890
Class: 6	0.03813869	0.03558394	0.03740876	0.9655586
Class: 7	0.04598540	0.04233577	0.04379562	0.9595523
Class: 8	0.03321168	0.02992701	0.03175182	0.9496057
Class: 9	0.03959854	0.03868613	0.04051095	0.9875292
Class: 10	0.03156934	0.02956204	0.03175182	0.9670775
Class: 11	0.03813869	0.03613139	0.03850365	0.9724510
Class: 12	0.02992701	0.02974453	0.03102190	0.9962928
Class: 13	0.03339416	0.03266423	0.03540146	0.9876551
Class: 14	0.03284672	0.03138686	0.03339416	0.9767400
Class: 15	0.03959854	0.03740876	0.03905109	0.9714952
Class: 16	0.04087591	0.03905109	0.04142336	0.9764419
Class: 17	0.04452555	0.04178832	0.04215328	0.9690713
Class: 18	0.04543796	0.04233577	0.04416058	0.9649076
Class: 19	0.03156934	0.03047445	0.03211679	0.9818110
Class: 20	0.03850365	0.03740876	0.03941606	0.9847381
Class: 21	0.03357664	0.03229927	0.03339416	0.9804118
Class: 22	0.03959854	0.03886861	0.04160584	0.9893584
Class: 23	0.03795620	0.03722628	0.03759124	0.9901949
Class: 24	0.04397810	0.04306569	0.04379562	0.9892448
Class: 25	0.04124088	0.04069343	0.04124088	0.9930773
Class: 26	0.03375912	0.03339416	0.03521898	0.9936503

Random Forest – 1000 Trees

```
> ##############################################################################
> #                        Random Forest 1000 Trees                            #
> ##############################################################################
> # Number of trees - 1000
> rf1000_model<-randomForest(Letter~.,data=trdf,ntree=1000)
> rf1000_pred<-predict(rf1000_model,tstdf[,-c(which(names(tstdf)=="Letter"))])
> rf1000_mtab<-table(tstdf$Letter,rf1000_pred)
> rf1000_cmx<-caret::confusionMatrix(rf1000_mtab)
> rf1000_mtab
```

Mohit Shankar Pursnani **Letter Recognition Dataset** mp5578@nyu.edu

Assignment - 3

```
   rf1000_pred
      1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22
1   222   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
2     0 198   0   1   0   0   1   1   0   0   0   0   0   0   0   0   0   4   0   0   1   0
3     0   0 211   0   2   1   5   0   0   0   0   1   0   0   1   0   0   0   0   0   1   0
4     0   2   0 238   0   0   0   2   0   0   0   0   0   2   4   0   0   0   0   0   0   0
5     0   0   0   0 211   0   7   0   0   0   2   0   0   0   0   0   1   0   0   1   0   0
6     0   1   0   0   0 196   0   0   0   1   0   0   0   0   0   5   0   0   0   2   0   0
7     0   0   1   1   1   0 233   0   0   0   0   0   0   0   0   1   0   2   1   0   0   0
8     0   0   0   1   0   1   1 162   0   0   3   0   0   0   1   0   1   4   0   0   0   0
9     0   0   0   1   0   0   0   0 210   8   0   0   0   0   0   2   0   0   0   1   0   0
10    0   0   0   0   1   1   0   0   7 163   1   0   0   0   1   0   0   0   0   0   0   0
11    0   1   0   0   1   0   0   1   0   0 200   0   1   0   0   0   0   4   0   0   1   0
12    0   0   0   0   2   0   1   0   0   0   0 162   0   0   0   0   2   2   0   0   0   0
13    0   2   0   0   1   0   0   0   0   0   1   0 184   1   0   1   1   0   0   0   1   1
14    0   1   0   0   0   0   0   3   0   0   0   0   2 174   2   0   0   0   0   0   0   0
15    0   1   1   1   0   0   0   0   0   0   0   0   0   0 205   0   5   1   0   0   0   0
16    0   1   0   0   0   8   1   1   0   0   0   0   0   0   0 215   0   0   0   0   0   1
17    0   0   0   0   0   0   1   0   0   0   0   0   0   0   1   0 227   1   0   0   0   0
18    0   6   0   2   0   0   0   0   0   0   0   0   0   2   0   1   0 230   0   0   0   0
19    0   1   0   0   2   1   0   2   0   0   0   0   0   0   0   0   1   1 168   0   0   0
20    0   1   2   0   0   1   0   0   0   0   1   0   0   0   0   0   0   0   2 207   0   0
21    0   0   0   0   0   0   0   1   0   0   0   0   3   0   0   0   2   0   0   0 177   0
22    0   7   0   0   0   0   0   0   0   0   0   0   0   2   1   2   0   0   0   0   0 214
23    0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
24    0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   1   1   0   0   0
25    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   0
26    0   2   0   0   0   2   0   0   0   2   0   0   0   0   0   0   0   4   0   0   0   0
   rf1000_pred
     23  24  25  26
1     0   0   0   0
2     0   2   0   0
3     0   0   0   0
4     0   0   0   0
5     0   1   0   1
6     0   0   0   0
7     0   0   0   0
8     0   0   0   0
9     0   0   0   0
10    0   0   0   0
11    0   2   0   0
12    0   1   0   0
13    1   0   0   0
14    1   0   0   0
15    0   0   0   0
16    0   0   0   0
17    0   0   0   1
18    0   1   0   0
19    0   0   0   0
20    0   0   2   0
21    0   0   0   0
22    1   0   1   0
23  204   0   0   0
24    0 237   0   0
25    0   0 224   0
26    0   0   0 183
```

Overall Statistics:

```
> rf1000_cmx$overall
      Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull AccuracyPValue
    0.95894161     0.95727727     0.95334681     0.96404068     0.04580292     0.00000000
 McnemarPValue
           NaN
```

Accuracy:

```
> (rf1000_accuracy<-sum(diag(rf1000_mtab))/sum(rf1000_mtab))
[1] 0.9589416
```

Statistics by Class:

```
> rf1000_cmx$byClass
          Sensitivity Specificity Pos Pred Value Neg Pred Value Precision    Recall
Class: 1    1.0000000   0.9998098      0.9955157      1.0000000 0.9955157 1.0000000
Class: 2    0.8839286   0.9980974      0.9519231      0.9950683 0.9519231 0.8839286
Class: 3    0.9813953   0.9979107      0.9504505      0.9992393 0.9504505 0.9813953
Class: 4    0.9714286   0.9980898      0.9596774      0.9986621 0.9596774 0.9714286
Class: 5    0.9547511   0.9975280      0.9419643      0.9980974 0.9419643 0.9547511
Class: 6    0.9289100   0.9982919      0.9560976      0.9971564 0.9560976 0.9289100
Class: 7    0.9282869   0.9986613      0.9708333      0.9965649 0.9708333 0.9282869
Class: 8    0.9364162   0.9977388      0.9310345      0.9979269 0.9310345 0.9364162
Class: 9    0.9677419   0.9977199      0.9459459      0.9986687 0.9459459 0.9677419
Class: 10   0.9367816   0.9979269      0.9367816      0.9979269 0.9367816 0.9367816
Class: 11   0.9523810   0.9979127      0.9478673      0.9981021 0.9478673 0.9523810
Class: 12   0.9938650   0.9984954      0.9529412      0.9998117 0.9529412 0.9938650
Class: 13   0.9684211   0.9981096      0.9484536      0.9988649 0.9484536 0.9684211
Class: 14   0.9613260   0.9983016      0.9508197      0.9986785 0.9508197 0.9613260
Class: 15   0.9490741   0.9982903      0.9579439      0.9979111 0.9579439 0.9490741
Class: 16   0.9471366   0.9977156      0.9471366      0.9977156 0.9471366 0.9471366
Class: 17   0.9265306   0.9992359      0.9826840      0.9965708 0.9826840 0.9265306
Class: 18   0.9163347   0.9977051      0.9504132      0.9959908 0.9504132 0.9163347
Class: 19   0.9824561   0.9984931      0.9545455      0.9994344 0.9545455 0.9824561
Class: 20   0.9764151   0.9982916      0.9583333      0.9990502 0.9583333 0.9764151
Class: 21   0.9725275   0.9988675      0.9672131      0.9990561 0.9672131 0.9725275
Class: 22   0.9907407   0.9973404      0.9385965      0.9996192 0.9385965 0.9907407
Class: 23   0.9855072   0.9996207      0.9902913      0.9994312 0.9902913 0.9855072
Class: 24   0.9713115   0.9994270      0.9875000      0.9986641 0.9875000 0.9713115
Class: 25   0.9867841   0.9996193      0.9911504      0.9994290 0.9911504 0.9867841
Class: 26   0.9891892   0.9981114      0.9481865      0.9996217 0.9481865 0.9891892
                 F1 Prevalence Detection Rate Detection Prevalence Balanced Accuracy
Class: 1  0.9977528 0.04051095     0.04051095           0.04069343         0.9999049
Class: 2  0.9166667 0.04087591     0.03613139           0.03795620         0.9410130
Class: 3  0.9656751 0.03923358     0.03850365           0.04051095         0.9896530
Class: 4  0.9655172 0.04470803     0.04343066           0.04525547         0.9847592
Class: 5  0.9483146 0.04032847     0.03850365           0.04087591         0.9761396
Class: 6  0.9423077 0.03850365     0.03576642           0.03740876         0.9636009
Class: 7  0.9490835 0.04580292     0.04251825           0.04379562         0.9634741
Class: 8  0.9337176 0.03156934     0.02956204           0.03175182         0.9670775
Class: 9  0.9567198 0.03959854     0.03832117           0.04051095         0.9827309
Class: 10 0.9367816 0.03175182     0.02974453           0.03175182         0.9673542
Class: 11 0.9501188 0.03832117     0.03649635           0.03850365         0.9751468
Class: 12 0.9729730 0.02974453     0.02956204           0.03102190         0.9961802
Class: 13 0.9583333 0.03467153     0.03357664           0.03540146         0.9832653
Class: 14 0.9560440 0.03302920     0.03175182           0.03339416         0.9798138
Class: 15 0.9534884 0.03941606     0.03740876           0.03905109         0.9736822
Class: 16 0.9471366 0.04142336     0.03923358           0.04142336         0.9724261
Class: 17 0.9537815 0.04470803     0.04142336           0.04215328         0.9628833
Class: 18 0.9330629 0.04580292     0.04197080           0.04416058         0.9570199
Class: 19 0.9682997 0.03120438     0.03065693           0.03211679         0.9904746
Class: 20 0.9672897 0.03868613     0.03777372           0.03941606         0.9873533
Class: 21 0.9698630 0.03321168     0.03229927           0.03339416         0.9856975
Class: 22 0.9639640 0.03941606     0.03905109           0.04160584         0.9940406
Class: 23 0.9878935 0.03777372     0.03722628           0.03759124         0.9925640
Class: 24 0.9793388 0.04452555     0.04324818           0.04379562         0.9853693
Class: 25 0.9889625 0.04142336     0.04087591           0.04124088         0.9932017
Class: 26 0.9682540 0.03375912     0.03339416           0.03521898         0.9936503
```

Boosting – GBM

Boosting is an ensemble learning method that combines a set of weak learners into a strong learner to minimize training errors. Gradient Boosting trains many models in a gradual, additive and sequential manner. One of the biggest motivations of using gradient boosting is that it allows one to optimize a user specified cost function, instead of a loss function that usually offers less control and does not essentially correspond with real world applications.

Mohit Shankar Pursnani **Letter Recognition Dataset** mp5578@nyu.edu

Assignment - 3

```
> ################################################################################
> #                              Boosting GBM                                    #
> ################################################################################
> trdf <- trainingDataset_withoutOutliers
> tstdf <- testingDataset_withoutOutliers
>
> gbm_table_3<-gbm(Letter~.,data=trdf,
+                  distribution="multinomial",
+                  n.trees=500,
+                  shrinkage=0.01,
+                  interaction.depth=3,
+                  n.minobsinnode=10,
+                  verbose=T,
+                  keep.data=F)
> labels = colnames(gbm_predict)[apply(gbm_predict, 1, which.max)]
>
> result = data.frame(tstdf$Letter, labels)
>
> cm = confusionMatrix(tstdf$Letter, as.factor(labels))
Warning message:
In confusionMatrix.default(tstdf$Letter, as.factor(labels)) :
  Levels are not in the same order for reference and data. Refactoring data to match.
>
> cm
Confusion Matrix and Statistics

          Reference
Prediction   1  10  11  12  13  14  15  16  17  18  19   2  20  21  22  23  24  25  26   3   4   5
        1  204   0   2   0   2   1   0   0   2   1   1   0   0   3   0   0   5   2   0   0   0   0
        10   0 141   0   1   0   1   2   2   2   0   1   4   0   1   0   0   2   0   0   0   3   0
        11   0   0 161   0   2   2   0   0   0  11   0   3   0   1   0   0   7   0   0   0   0   5
        12   0   0   1 136   0   0   0   0   4   2   2   2   1   0   0   0   4   1   0   4   0   7
        13   1   0   4   0 166   7   2   0   0   1   0   2   0   4   0   1   0   0   0   1   1   0
        14   0   0   1   0   0 159   3   0   0   3   0   1   1   0   0   3   0   2   0   1   5   0
        15   0   0   0   0   0   0 180   0  10   3   0   1   0   0   0   6   0   0   0   0   5   0
        16   0   0   1   0   0   0   2 185   1   0   1   3   0   0   1   0   0   6   0   0   3   2
        17   0   0   0   2   0   0  18   0 187   0   0   7   0   0   0   0   3   0   6   1   0   4
        18   0   0   2   0   1   1   1   0   1 204   0  14   0   2   0   0   3   0   0   0   6   4
        19   0   2   0   0   0   2   0   0   4   7 137   3   3   0   0   0   3   0   4   0   1   4
        2    0   0   1   0   1   2   1   0   3   9   3 175   0   0   1   0   6   0   0   0   3   0
        20   0   1   2   0   0   1   2   3   0   0   4   1 172   5   4   0   2   0   3   0   0   5
        21   0   0   2   0   5   3   4   0   3   0   0   0   0 163   0   0   0   1   0   1   0   0
        22   0   0   0   0   0   3   0   3   3   0   0   4   1   3 196   4   1   6   0   0   0   0
        23   0   0   0   0   4   1   5   0   1   1   0   0   0   1   3 188   0   0   0   0   0   0
        24   0   0   5   0   0   0   1   0   0   2   4   6   2   0   1   0 212   1   0   0   3   1
        25   0   0   0   0   0   0   3   1   9   0   1   0   1   4   2   1   1 197   1   0   1   0
        26   0   5   0   1   0   0   0   0   3   2  12   5   2   0   0   0   1   0 153   0   1   6
        3    0   0   3   0   1   1   4   0   0   0   2   0   1   6   0   0   1   0   0 186   0  10
        4    0   1   0   0   0   1   6   1   0   4   0  15   2   0   0   0   1   0   0   0 204   1
        5    0   0   7   0   0   0   0   0   8   0   4   0   0   0   0   0   5   0   4   5   0 178
        6    0   0   0   0   0   0   1   0   3   0   1   4   6   1   0   1   0   0   1   0   4   2
        7    1   0   2   0   2   0   1   1   8   2   1   6   0   0   0   9   0   0   0  11   3   3
        8    0   1   4   0   1   0   5   2   1  10   0   5   0   2   2   2   2   2   0   0  10   1
        9    0   2   0   0   0   0   0   1   1   0   4   2   0   0   0   0   5   0   0   2   2   0

          Reference
Prediction   6   7   8   9
        1    0   0   0   0
        10   3   0   4   7
        11   1   2  16   0
        12   0   6   0   0
        13   2   1   1   0
        14   0   0   4   0
        15   0   5   4   0
        16  16   5   1   0
        17   0   3   0   0
        18   0   1   2   0
        19   4   0   2   0
        2    1   0   2   0
        20   9   0   2   0
        21   0   0   1   0
        22   0   1   3   0
        23   0   0   2   0
        24   0   0   0   2
        25   3   0   1   0
        26   2   0   0   0
        3    1   6   0   0
        4    1   2   9   0
        5    0  13   0   0
        6  181   0   0   0
        7    0 185   5   0
        8    1   3 120   0
        9    4   1   2 196
```

Mohit Shankar Pursnani **Letter Recognition Dataset** **mp5578@nyu.edu**
Assignment - 3

```
Overall Statistics

               Accuracy : 0.8332
                 95% CI : (0.8231, 0.843)
    No Information Rate : 0.0484
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8264

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 10 Class: 11 Class: 12 Class: 13 Class: 14 Class: 15 Class: 16
Sensitivity           0.99029   0.92157   0.81313   0.97143   0.89730   0.85484   0.75000   0.91584
Specificity           0.99640   0.99381   0.99053   0.99363   0.99471   0.99547   0.99351   0.99204
Pos Pred Value        0.91480   0.81034   0.76303   0.80000   0.85567   0.86885   0.84112   0.81498
Neg Pred Value        0.99962   0.99774   0.99298   0.99925   0.99641   0.99490   0.98861   0.99676
Prevalence            0.03759   0.02792   0.03613   0.02555   0.03376   0.03394   0.04380   0.03686
Detection Rate        0.03723   0.02573   0.02938   0.02482   0.03029   0.02901   0.03285   0.03376
Detection Prevalence  0.04069   0.03175   0.03850   0.03102   0.03540   0.03339   0.03905   0.04142
Balanced Accuracy     0.99334   0.95769   0.90183   0.98253   0.94600   0.92515   0.87176   0.95394
                     Class: 17 Class: 18 Class: 19 Class: 2 Class: 20 Class: 21 Class: 22 Class: 23
Sensitivity           0.74502   0.77567   0.75691  0.66038   0.91979   0.83590   0.92891   0.87850
Specificity           0.99159   0.99272   0.99264  0.99367   0.99169   0.99622   0.99393   0.99658
Pos Pred Value        0.80952   0.84298   0.77841  0.84135   0.79630   0.89071   0.85965   0.91262
Neg Pred Value        0.98781   0.98874   0.99170  0.98293   0.99715   0.99396   0.99714   0.99507
Prevalence            0.04580   0.04799   0.03303  0.04836   0.03412   0.03558   0.03850   0.03905
Detection Rate        0.03412   0.03723   0.02500  0.03193   0.03139   0.02974   0.03577   0.03431
Detection Prevalence  0.04215   0.04416   0.03212  0.03796   0.03942   0.03339   0.04161   0.03759
Balanced Accuracy     0.86830   0.88419   0.87477  0.82702   0.95574   0.91606   0.96142   0.93754
                     Class: 24 Class: 25 Class: 26 Class: 3 Class: 4 Class: 5 Class: 6 Class: 7
Sensitivity           0.80303   0.89954   0.89474  0.87736  0.80000  0.76395  0.79039  0.79060
Specificity           0.99463   0.99449   0.99247  0.99317  0.99158  0.99123  0.99543  0.98952
Pos Pred Value        0.88333   0.87168   0.79275  0.83784  0.82258  0.79464  0.88293  0.77083
Neg Pred Value        0.99008   0.99581   0.99660  0.99506  0.99025  0.98954  0.99090  0.99065
Prevalence            0.04818   0.03996   0.03120  0.03869  0.04653  0.04252  0.04179  0.04270
Detection Rate        0.03869   0.03595   0.02792  0.03394  0.03723  0.03248  0.03303  0.03376
Detection Prevalence  0.04380   0.04124   0.03522  0.04051  0.04526  0.04088  0.03741  0.04380
Balanced Accuracy     0.89883   0.94702   0.94360  0.93526  0.89579  0.87759  0.89291  0.89006
                     Class: 8 Class: 9
Sensitivity           0.66298  0.95610
Specificity           0.98981  0.99507
Pos Pred Value        0.68966  0.88288
Neg Pred Value        0.98850  0.99829
Prevalence            0.03303  0.03741
Detection Rate        0.02190  0.03577
Detection Prevalence  0.03175  0.04051
Balanced Accuracy     0.82640  0.97558
```

Observations:

Both **bias and variance are reduced by Cross Validation**.

We observed that **Cross Validation on LDA, QDA, Tree and SVM is not suitable** as it results in overfitting. Judging accuracy based on just training set will be wrong

Tree algorithm is not suitable for this dataset as with more class labels the prediction becomes more complex. Decision tree algorithm while training uses only some of the features as Decision Tree is immune to multicollinearity, while building model it considers only few of the feature variables which is not suitable for this dataset.

Random Forest on the other hand operates by constructing a multitude of decision trees at training time. Specially during classification, the output of the random forest is the class selected by most trees. Hence **random forest gives high accuracy**.

We saw that **bagging helped in improving the accuracy** of the model rather than CV and reducing the variance of the models and this helped us in eliminating the challenge of overfitting. Bagging usually aggregates the predictions to select the best prediction. We also saw that the accuracy of Bagging with the Tree model was better than that we got in Assignment 2, this is because Bagging with Decision Tree tries to reduce the problem of overfitting, but still the accuracy was not very good even after applying bagging because tree

model is not able to correctly predict in the learning and Generalization phase. The tree model is underfitting the dataset.
Boosting combines a set of weak learners into a strong learner to minimize training errors hence the accuracy using Bagging GBM is 0.8332.

The **accuracy of the KNN & Random Forest overall for this dataset is better than most of the other algorithms or the ensemble techniques that we applied**. The same we observed when we used KNN without CV, with CV and with bagging.
We also observed that **Sensitivity** – True Positive Detection, **Specificity** – True Negative Detection, the **Positive Predictive** value - proportion of predicted positives which are actual positives, the **Negative Predictive** value - proportion of predicted negatives which are actual negatives, **Precision**– the quality of a positive prediction, **Recall** - how many of the correct hits were found are better for KNN model with CV and without CV, KNN with Bagging & for Random Forest than any of the other models we have applied and hence overall accuracy of these algorithm is high. If there is a need to choose to choose some algorithms for this dataset based on the observations that we have seen choosing KNN or Random Forest is the most appropriate option.

Overall, these assignments have given a good start to us, it has helped us in understanding the behavior of the algorithms and the behavior of ensemble techniques from a practical stand-point rather than just theoretical reading.