Letter Image Recognition

Goal of this project is to compare different classifier systems to learn to correctly guess the letter categories associated with feature of 16 integer attributes extracted from scan images of the letters.

The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet.

Attribute Information:

1. lettr capital letter (26 values from A to Z)

2. x-box horizontal position of box (integer)

3. y-box vertical position of box (integer)

4. width width of box (integer)

5. high height of box (integer)

6. onpix total # on pixels (integer)

7. x-bar mean x of on pixels in box (integer)

8. y-bar mean y of on pixels in box (integer)

9. x2bar mean x variance (integer)

10. y2bar mean y variance (integer)

11. xybar mean x y correlation (integer)

12. x2ybr mean of x \* x \* y (integer)

13. xy2br mean of x \* y \* y (integer)

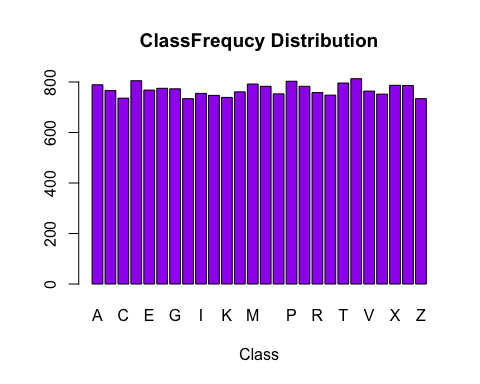
14. x-ege mean edge count left to right (integer)

15. xegvy correlation of x-ege with y (integer)

16. y-ege mean edge count bottom to top (integer)

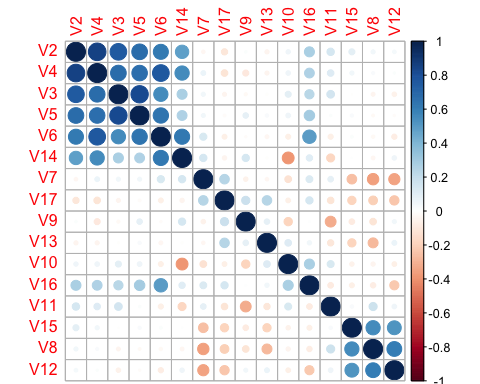
17. yegvx correlation of y-ege with x (integer)

After loading the data, class distributions were observed to see any class imbalances. As the figure below shows class frequency distribution is uniform.



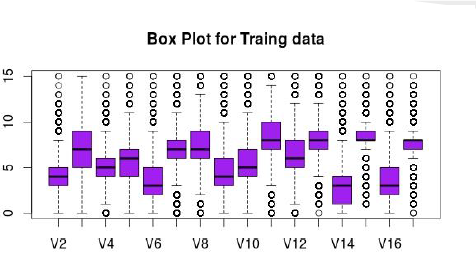
Preprocessing the data set:

Correlations between predictors were visualized to find out if there are correlations greater than 0.75. As the figure below shows there are some predicators, which are correlated. These high correlated predictors were removed using the caret packages, findCorrelation() function.



After removing the high correlated predictors, box plots were used to find out skewness in predictor variables.

The figure below shows boxplots for all the predictors and none of them show too high or low skewness. So transformations of data weren’t necessary.



Model building :

Linear Models

1. Linear Discriminant Analysis

15000 samples  
 13 predictor  
 26 classes: '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'   
   
 Pre-processing: centered, scaled   
 Resampling: Bootstrapped (25 reps)   
   
 Summary of sample sizes: 15000, 15000, 15000, 15000, 15000, 15000, ...   
   
 Resampling results across tuning parameters:  
   
 dimen Accuracy Kappa Accuracy SD Kappa SD   
 1 0.1685297 0.1355126 0.008434755 0.008705092  
 2 0.3497718 0.3236751 0.006712291 0.006945867  
 3 0.4289124 0.4060080 0.015855319 0.016455423  
 4 0.5390830 0.5205938 0.007712305 0.008013558  
 5 0.5833730 0.5666671 0.006934202 0.007205122  
 6 0.6151006 0.5996624 0.008165075 0.008487258  
 7 0.6250795 0.6100264 0.009064244 0.009421349  
 8 0.6726983 0.6595546 0.006922812 0.007192980  
 9 0.6816396 0.6688550 0.006811382 0.007074126  
 10 0.6904524 0.6780292 0.006649750 0.006907253  
 11 0.6867831 0.6742183 0.007103359 0.007377300  
 12 0.6925854 0.6802618 0.007167037 0.007444857  
 13 0.6915845 0.6792207 0.006972897 0.007243935  
   
 Kappa was used to select the optimal model using the largest value.  
 The final value used for the model was dimen = 12.

Confusion Matrix and Statistics for testing data  
  
 Overall Statistics  
   
 Accuracy : 0.6882   
 Kappa : 0.6757   
   
2 . Partial Least Square Discrimination Analysis   
   
 15000 samples  
 13 predictor  
 26 classes: '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'   
   
 Pre-processing: centered, scaled   
 Resampling: Bootstrapped (25 reps)   
   
 Summary of sample sizes: 15000, 15000, 15000, 15000, 15000, 15000, ...   
   
 Resampling results across tuning parameters:  
   
 ncomp Accuracy Kappa Accuracy SD Kappa SD   
 1 0.07420849 0.0372415 0.003271222 0.002700414  
 2 0.17892449 0.1469015 0.010474731 0.010607809  
   
 Kappa was used to select the optimal model using the largest value.  
 The final value used for the model was ncomp = 2.

Overall Statistics  
   
 Accuracy : 0.1908   
 Kappa : 0.1571

Non - Linear models

1. K Nearest Neighbor Classification

15000 samples

13 predictor

26 classes: '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'

Pre-processing: centered, scaled

Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 15000, 15000, 15000, 15000, 15000, 15000, ...

Resampling results across tuning parameters:

k Accuracy Kappa Accuracy SD Kappa SD

3 0.9199277 0.9167123 0.004078469 0.004242216

5 0.9188687 0.9156107 0.003562830 0.003706130

7 0.9178972 0.9146001 0.004523367 0.004705213

9 0.9168892 0.9135518 0.004898951 0.005096635

11 0.9137899 0.9103282 0.004784319 0.004977109

Kappa was used to select the optimal model using the largest value.

The final value used for the model was k = 3.

myknn<-knn(trainX,testX,as.factor(trainY), k = 3, l = 0, prob = FALSE, use.all = TRUE)

confusionMatrix(myknn,testY)  
 Overall Statistics  
   
 Accuracy : 0.9544   
 Kappa : 0.9526

Improved Knn. For the problem of nearest neighbor classification, a simpler approach called "leave-out-one" cross-validation can be used, and this is provided by the knn.cv function. Using this technique, the observation itself is ignored when looking for its neighbors.

2. KNN.CV

myknn<-knn.cv(x, y, k = 3, l = 0, prob = FALSE, use.all = TRUE)

confusionMatrix(myknn, y)

Overall Statistics  
   
 Accuracy : 0.9614   
 Kappa : 0.9599