Homework 7

By Shivani Sanjay Mahaddalkar

## Introduction

The analysis aims to build models to recognize the handwritten digits from 0-9. The data set comes from the Kaggle Digit Recognizer competition. We build k nearest neighbor, support vector machine and random forest algorithms and compare their accuracies.

1. **Data Processing**

Importing appropriate libraries

require(caret)

require(e1071)

require(rpart)

require(dplyr)

require(stringr)

require(randomForest)

library(tidyverse)

library(kernlab)

library(e1071)

Setting directory from which to extract data

setwd("D:/Spring2021/DataAnalytics")  
digit\_train <- read.csv("digit-train.csv")  
digit\_test <- read.csv("digit-test.csv")

Data Transformation: The label column which is the prediction column or the target column is changed from an integer type to a factor type to make the model understand that classification with 10 levels from 0-9 is taking place.

To increase the speed of the algorithms, the columns which show zero variance are dropped. Losing these columns does not impact the prediction as they have no variance.

var0 <- apply(digit\_train, 2, var)!=0  
digit\_train <- digit\_train[,var0]  
  
digit\_test <- digit\_test[,colnames(digit\_train)]  
digit\_train$label <- factor(paste0('X', digit\_train$label), levels = c('X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8', 'X9'))  
digit\_test$label <- factor(paste0('X', digit\_test$label), levels = c('X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8', 'X9'))

## Model: KNN algorithm

For KNN algorithm we search through 4 different values of k.

On running cross validation on the training set, we find that k =5 shows the top most accuracy of 92.26%.

When the k=5 KNN model is run on the testing dataset it shows an accuracy of 93.47%. On further inspection of the confusion matrix, we can see that the algorithm does not mistake two very different looking digits. A lot of 4s are predicted to be 9s, 8s are predicted to be 5s.

We can say that the model shows good accuracy.

search\_grid = expand.grid(k = c(5, 7, 9, 11))  
  
# set up 3-fold cross validation procedure  
train\_control <- trainControl(  
 method = "cv",   
 number = 3  
 )  
  
# more advanced option, run 3 fold cross validation 5 times  
train\_control\_adv <- trainControl(  
 method = "repeatedcv",   
 number = 3,  
 repeats = 5  
 )  
  
# train model  
knn <- train(label ~ .,  
 data = digit\_train,  
 method = "knn",  
 trControl = train\_control\_adv,  
 tuneGrid = search\_grid  
 )

# top model  
  
knn$results %>%   
 top\_n(1, wt = Accuracy) %>%  
 arrange(desc(Accuracy))

## k Accuracy Kappa AccuracySD KappaSD  
## 1 5 0.9226282 0.9139844 0.007255681 0.008066505

# results for best model  
confusionMatrix(knn)

## Cross-Validated (3 fold, repeated 5 times) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction X0 X1 X2 X3 X4 X5 X6 X7 X8 X9  
## X0 9.7 0.0 0.1 0.0 0.0 0.0 0.1 0.0 0.1 0.1  
## X1 0.0 11.2 0.3 0.1 0.2 0.1 0.1 0.4 0.4 0.1  
## X2 0.0 0.0 8.7 0.2 0.0 0.0 0.0 0.0 0.0 0.0  
## X3 0.0 0.0 0.1 9.2 0.0 0.3 0.0 0.0 0.3 0.1  
## X4 0.0 0.0 0.1 0.0 9.0 0.0 0.0 0.1 0.1 0.3  
## X5 0.0 0.0 0.0 0.3 0.0 8.5 0.1 0.0 0.4 0.0  
## X6 0.1 0.0 0.1 0.1 0.1 0.2 9.6 0.0 0.1 0.0  
## X7 0.0 0.0 0.4 0.1 0.0 0.0 0.0 9.7 0.1 0.3  
## X8 0.0 0.0 0.1 0.2 0.0 0.0 0.0 0.0 8.8 0.1  
## X9 0.0 0.0 0.0 0.0 0.6 0.1 0.0 0.2 0.3 8.0  
##   
## Accuracy (average) : 0.9226

pred <- predict(knn, newdata = digit\_test)

confusionMatrix(pred, digit\_test$label)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction X0 X1 X2 X3 X4 X5 X6 X7 X8 X9  
## X0 408 0 7 2 0 1 6 0 3 1  
## X1 0 474 13 8 8 5 1 12 11 3  
## X2 0 2 384 1 0 0 0 0 0 0  
## X3 0 0 3 413 0 9 0 0 9 8  
## X4 0 0 0 0 366 0 1 4 4 8  
## X5 1 0 1 5 0 363 4 0 18 1  
## X6 4 1 1 2 1 6 392 0 4 1  
## X7 0 1 7 8 2 0 0 434 1 11  
## X8 1 0 3 3 0 1 0 0 337 0  
## X9 0 0 1 4 27 3 0 15 6 353  
##   
## Overall Statistics  
##   
## Accuracy : 0.9347   
## 95% CI : (0.9268, 0.942)  
## No Information Rate : 0.1139   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9274   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: X0 Class: X1 Class: X2 Class: X3 Class: X4  
## Sensitivity 0.98551 0.9916 0.91429 0.92601 0.90594  
## Specificity 0.99471 0.9836 0.99921 0.99227 0.99552  
## Pos Pred Value 0.95327 0.8860 0.99225 0.93439 0.95561  
## Neg Pred Value 0.99841 0.9989 0.99055 0.99121 0.99004  
## Prevalence 0.09862 0.1139 0.10005 0.10624 0.09624  
## Detection Rate 0.09719 0.1129 0.09147 0.09838 0.08718  
## Detection Prevalence 0.10195 0.1274 0.09219 0.10529 0.09123  
## Balanced Accuracy 0.99011 0.9876 0.95675 0.95914 0.95073  
## Class: X5 Class: X6 Class: X7 Class: X8 Class: X9  
## Sensitivity 0.93557 0.97030 0.9333 0.85751 0.91451  
## Specificity 0.99213 0.99473 0.9920 0.99790 0.98531  
## Pos Pred Value 0.92366 0.95146 0.9353 0.97681 0.86308  
## Neg Pred Value 0.99343 0.99683 0.9917 0.98547 0.99129  
## Prevalence 0.09242 0.09624 0.1108 0.09362 0.09195  
## Detection Rate 0.08647 0.09338 0.1034 0.08028 0.08409  
## Detection Prevalence 0.09362 0.09814 0.1105 0.08218 0.09743  
## Balanced Accuracy 0.96385 0.98251 0.9626 0.92770 0.94991

1. **Model: SVM**

For SVM algorithm we check for linear and radial kernel.

We find that linear svm model has an accuracy of 90.74%, whereas the radial svm model shows and accuracy of 91.305% on cross validating the dataset.

When both the models are run on the testing dataset linear model shows an accuracy of 91.42 %, whereas radial model shows an accuracy of 92.33%. Both of the models, work well however, radial model works slightly better. However, while computing the radial model takes significantly longer to train, therefore, the tradeoff between computing resources and accuracy was visible during this part of the analysis.

We can say that the model shows good accuracy.

set.seed(123)  
svm.m1 <- svm(label ~ ., data = digit\_train, type = 'C', kernel = 'linear', cross = 3, probability = TRUE)  
summary(svm.m1)

##   
## Call:  
## svm(formula = label ~ ., data = digit\_train, type = "C", kernel = "linear",   
## cross = 3, probability = TRUE)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: linear   
## cost: 1   
##   
## Number of Support Vectors: 1572  
##   
## ( 171 200 171 115 138 141 207 165 180 84 )  
##   
##   
## Number of Classes: 10   
##   
## Levels:   
## X0 X1 X2 X3 X4 X5 X6 X7 X8 X9  
##   
## 3-fold cross-validation on training data:  
##   
## Total Accuracy: 90.73368   
## Single Accuracies:  
## 90.42173 90.92209 90.85714

set.seed(123)  
svm.m2 <- svm(label ~ ., data = digit\_train, type = 'C', kernel = 'radial', cross = 3, probability = TRUE)  
summary(svm.m2)

##   
## Call:  
## svm(formula = label ~ ., data = digit\_train, type = "C", kernel = "radial",   
## cross = 3, probability = TRUE)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 1   
##   
## Number of Support Vectors: 2425  
##   
## ( 279 294 287 175 226 265 273 254 254 118 )  
##   
##   
## Number of Classes: 10   
##   
## Levels:   
## X0 X1 X2 X3 X4 X5 X6 X7 X8 X9  
##   
## 3-fold cross-validation on training data:  
##   
## Total Accuracy: 91.30538   
## Single Accuracies:  
## 91.42244 91.77984 90.71429

pred1 <- predict(svm.m1, newdata = digit\_test)  
pred2 <- predict(svm.m2, newdata = digit\_test)

confusionMatrix(pred1, digit\_test$label)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction X0 X1 X2 X3 X4 X5 X6 X7 X8 X9  
## X0 403 0 3 4 0 0 5 0 0 2  
## X1 0 468 5 6 2 2 0 2 3 1  
## X2 0 3 380 11 5 3 10 3 10 1  
## X3 1 3 15 395 0 23 1 2 13 5  
## X4 2 0 4 0 374 0 2 13 1 15  
## X5 3 1 4 12 0 339 5 1 21 2  
## X6 1 0 2 1 2 7 378 0 3 0  
## X7 1 0 1 6 3 0 0 430 1 24  
## X8 3 2 5 11 0 10 3 0 338 3  
## X9 0 1 1 0 18 4 0 14 3 333  
##   
## Overall Statistics  
##   
## Accuracy : 0.9142   
## 95% CI : (0.9054, 0.9225)  
## No Information Rate : 0.1139   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9046   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: X0 Class: X1 Class: X2 Class: X3 Class: X4  
## Sensitivity 0.97343 0.9791 0.90476 0.88565 0.92574  
## Specificity 0.99630 0.9944 0.98782 0.98321 0.99025  
## Pos Pred Value 0.96643 0.9571 0.89202 0.86245 0.90998  
## Neg Pred Value 0.99709 0.9973 0.98940 0.98636 0.99208  
## Prevalence 0.09862 0.1139 0.10005 0.10624 0.09624  
## Detection Rate 0.09600 0.1115 0.09052 0.09409 0.08909  
## Detection Prevalence 0.09933 0.1165 0.10148 0.10910 0.09790  
## Balanced Accuracy 0.98487 0.9867 0.94629 0.93443 0.95800  
## Class: X5 Class: X6 Class: X7 Class: X8 Class: X9  
## Sensitivity 0.87371 0.93564 0.9247 0.86005 0.86269  
## Specificity 0.98714 0.99578 0.9904 0.99028 0.98924  
## Pos Pred Value 0.87371 0.95939 0.9227 0.90133 0.89037  
## Neg Pred Value 0.98714 0.99317 0.9906 0.98561 0.98614  
## Prevalence 0.09242 0.09624 0.1108 0.09362 0.09195  
## Detection Rate 0.08075 0.09004 0.1024 0.08051 0.07932  
## Detection Prevalence 0.09242 0.09385 0.1110 0.08933 0.08909  
## Balanced Accuracy 0.93043 0.96571 0.9575 0.92516 0.92597

confusionMatrix(pred2, digit\_test$label)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction X0 X1 X2 X3 X4 X5 X6 X7 X8 X9  
## X0 403 0 2 3 0 0 5 0 1 1  
## X1 0 465 3 4 1 2 0 4 6 2  
## X2 2 7 392 17 8 3 2 1 7 4  
## X3 0 2 2 397 0 13 1 1 8 6  
## X4 0 0 3 1 374 0 2 7 1 13  
## X5 1 1 0 6 0 356 9 1 10 1  
## X6 4 0 3 1 0 7 366 0 3 0  
## X7 1 2 9 11 8 4 17 434 3 23  
## X8 2 1 6 6 1 2 2 0 354 1  
## X9 1 0 0 0 12 1 0 17 0 335  
##   
## Overall Statistics  
##   
## Accuracy : 0.9233   
## 95% CI : (0.9148, 0.9312)  
## No Information Rate : 0.1139   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9147   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: X0 Class: X1 Class: X2 Class: X3 Class: X4  
## Sensitivity 0.97343 0.9728 0.93333 0.89013 0.92574  
## Specificity 0.99683 0.9941 0.98650 0.99120 0.99288  
## Pos Pred Value 0.97108 0.9548 0.88488 0.92326 0.93267  
## Neg Pred Value 0.99709 0.9965 0.99254 0.98700 0.99210  
## Prevalence 0.09862 0.1139 0.10005 0.10624 0.09624  
## Detection Rate 0.09600 0.1108 0.09338 0.09457 0.08909  
## Detection Prevalence 0.09886 0.1160 0.10553 0.10243 0.09552  
## Balanced Accuracy 0.98513 0.9834 0.95992 0.94067 0.95931  
## Class: X5 Class: X6 Class: X7 Class: X8 Class: X9  
## Sensitivity 0.91753 0.90594 0.9333 0.90076 0.86788  
## Specificity 0.99239 0.99526 0.9791 0.99448 0.99187  
## Pos Pred Value 0.92468 0.95313 0.8477 0.94400 0.91530  
## Neg Pred Value 0.99161 0.99004 0.9916 0.98980 0.98669  
## Prevalence 0.09242 0.09624 0.1108 0.09362 0.09195  
## Detection Rate 0.08480 0.08718 0.1034 0.08433 0.07980  
## Detection Prevalence 0.09171 0.09147 0.1220 0.08933 0.08718  
## Balanced Accuracy 0.95496 0.95060 0.9562 0.94762 0.92987

1. **Random Forest**

For random forest algorithm we conduct a 3 fold cross validation.

We find that linear svm model has an accuracy of 93.27% on cross validating the dataset.

When both the model is run on the testing dataset it shows an accuracy of 94.07%. This computation took the longest to run, even though the parameters were in a limited capacity.

We can say that the model shows good accuracy.

set.seed(123)  
train\_control <- trainControl(method = 'repeatedcv', number = 3, repeats = 3)  
rf <- train(label ~ ., data = digit\_train, method = 'rf', metric = 'Accuracy', trControl = train\_control, type = 'C')  
confusionMatrix(rf)

## Cross-Validated (3 fold, repeated 3 times) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction X0 X1 X2 X3 X4 X5 X6 X7 X8 X9  
## X0 9.7 0.0 0.0 0.1 0.0 0.1 0.1 0.0 0.1 0.1  
## X1 0.0 11.0 0.1 0.0 0.0 0.1 0.0 0.1 0.1 0.0  
## X2 0.0 0.0 9.0 0.3 0.1 0.0 0.0 0.2 0.1 0.0  
## X3 0.0 0.0 0.1 9.1 0.0 0.3 0.0 0.0 0.3 0.1  
## X4 0.0 0.0 0.1 0.0 9.5 0.1 0.1 0.1 0.1 0.2  
## X5 0.0 0.0 0.0 0.3 0.0 8.5 0.2 0.0 0.1 0.1  
## X6 0.0 0.0 0.1 0.1 0.1 0.1 9.4 0.0 0.0 0.0  
## X7 0.0 0.0 0.2 0.1 0.0 0.0 0.0 9.5 0.0 0.1  
## X8 0.1 0.1 0.1 0.2 0.0 0.1 0.1 0.0 9.4 0.1  
## X9 0.0 0.0 0.0 0.0 0.3 0.0 0.0 0.3 0.2 8.1  
##   
## Accuracy (average) : 0.9327

pred3 <- predict(rf, newdata = digit\_test)  
confusionMatrix(pred3, digit\_test$label)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction X0 X1 X2 X3 X4 X5 X6 X7 X8 X9  
## X0 407 0 1 3 1 3 3 3 2 4  
## X1 0 468 0 5 1 4 0 5 4 2  
## X2 1 5 405 8 2 1 1 3 4 0  
## X3 1 1 3 404 0 8 0 0 13 8  
## X4 1 0 3 0 378 0 2 8 1 9  
## X5 1 1 1 5 0 362 5 0 8 1  
## X6 0 0 2 4 2 3 391 0 2 1  
## X7 0 2 3 8 0 0 0 433 0 9  
## X8 3 0 2 6 3 3 2 1 355 6  
## X9 0 1 0 3 17 4 0 12 4 346  
##   
## Overall Statistics  
##   
## Accuracy : 0.9407   
## 95% CI : (0.9331, 0.9476)  
## No Information Rate : 0.1139   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9341   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: X0 Class: X1 Class: X2 Class: X3 Class: X4  
## Sensitivity 0.98309 0.9791 0.96429 0.90583 0.93564  
## Specificity 0.99471 0.9944 0.99338 0.99094 0.99367  
## Pos Pred Value 0.95316 0.9571 0.94186 0.92237 0.94030  
## Neg Pred Value 0.99814 0.9973 0.99602 0.98883 0.99315  
## Prevalence 0.09862 0.1139 0.10005 0.10624 0.09624  
## Detection Rate 0.09695 0.1115 0.09647 0.09624 0.09004  
## Detection Prevalence 0.10172 0.1165 0.10243 0.10434 0.09576  
## Balanced Accuracy 0.98890 0.9867 0.97883 0.94838 0.96466  
## Class: X5 Class: X6 Class: X7 Class: X8 Class: X9  
## Sensitivity 0.93299 0.96782 0.9312 0.90331 0.89637  
## Specificity 0.99423 0.99631 0.9941 0.99317 0.98924  
## Pos Pred Value 0.94271 0.96543 0.9516 0.93176 0.89406  
## Neg Pred Value 0.99318 0.99657 0.9915 0.99004 0.98950  
## Prevalence 0.09242 0.09624 0.1108 0.09362 0.09195  
## Detection Rate 0.08623 0.09314 0.1031 0.08456 0.08242  
## Detection Prevalence 0.09147 0.09647 0.1084 0.09076 0.09219  
## Balanced Accuracy 0.96361 0.98207 0.9626 0.94824 0.94281

1. **Conclusion**

For different models on the same training and testing dataset we found varying accuracies. The best accuracies for these models are given in the table below.

|  |  |
| --- | --- |
| Model | Accuracy |
| Decision Tree | 60.7% |
| Naïve Bayes classifier | 46.36% |
| KNN | 93.47% |
| SVM | 92.33% |
| Random Forest | 94.07% |

The models KNN, SVM and random forest have accuracy greater than 90%. Random forest showed the best accuracy out of the 5. With more computing ability, the features could be engineered further to get even better results.