DIGIT RECOGNIZER IN R USING MNIST DATASET

AMAL THOMAS

12/11/2021

Digit Recognizer in R

Introduction

Using the Mnist Dataset for doing the Digit recognition. We have got the ‘train.csv’ and ‘test.csv’ files from the kaggle wedsite. In the ‘train.csv’ contains 785 columns and 42000 rows and the ‘test.csv’ contains 28000 rows and 784 columns. Here we using the Random Forest algorithm for the Digit recognition problem.

Table of Contents

- 1. Data understanding
- 2. Model building and prediction
- 3. Conclusion

Data understanding

As always, first I read in data and take a look at it. The `train.csv` has one digit label column and 784 columns with pixel color values that go from 0 to 255.

```
rm(list = ls())

#loading libraries
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.5
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.0.5
```

```
#loading datasets
train <- fread("D:/SEM III/Predictive analytics/train.csv")
test <- fread("D:/SEM III/Predictive analytics/test.csv")
```

```
##Rows and Columns
dim(train)
```

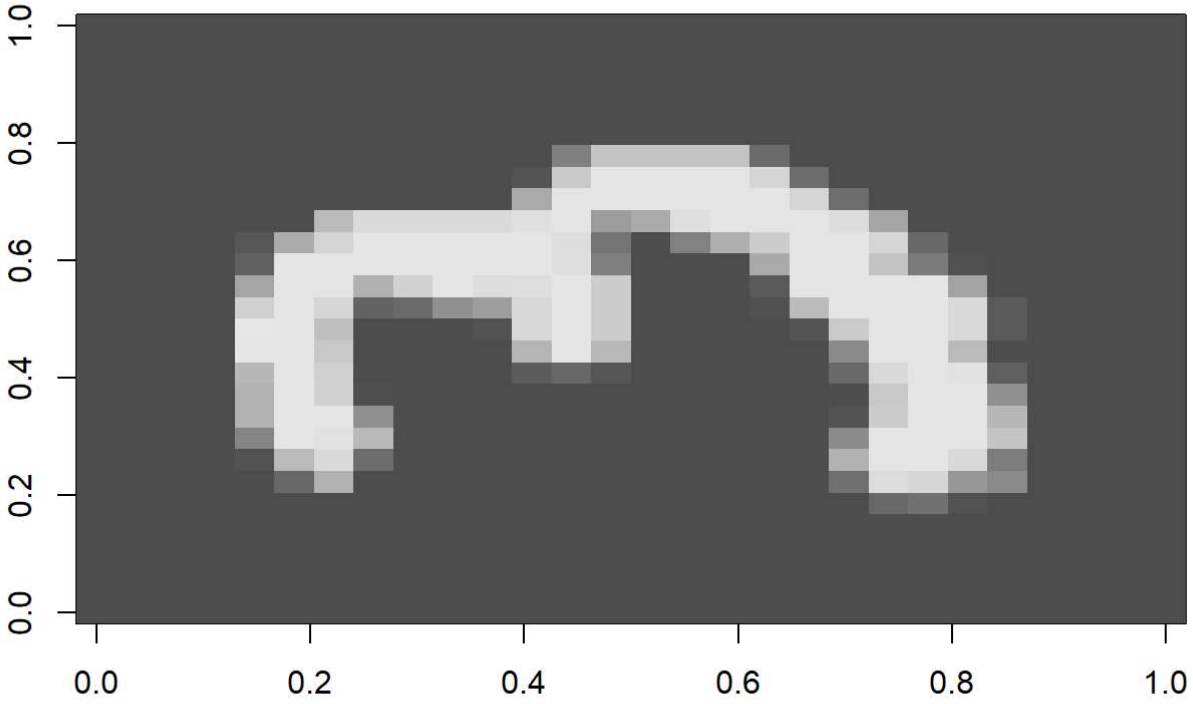
```
## [1] 42000   785
```

```
dim(test)
```

```
## [1] 28000   784
```

##Just only checcking the 10th observation , is it showing correct or not.

```
m = matrix(unlist(train[10,-1]),nrow = 28,byrow = T)
# Plot that matrix
image(m,col=grey.colors(255))
```



Model building and prediction

```
#making RF model
set.seed(123)
train$label <- factor(train$label)
rf_model <- randomForest(data = train, label ~ ., ntree = 20, do.trace = 1)
```

```
## ntree      OOB      1      2      3      4      5      6      7      8      9     10
##     1:  20.15% 11.92%  6.26% 22.86% 27.60% 18.60% 30.98% 16.66% 16.57% 27.94% 25.03%
##     2:  20.02% 11.48%  5.63% 22.12% 26.89% 19.56% 28.51% 17.05% 15.90% 30.64% 25.10%
##     3:  19.15% 10.08%  5.21% 21.01% 25.05% 18.50% 28.29% 16.60% 14.90% 29.85% 24.70%
##     4:  17.98%  9.38%  4.74% 19.62% 22.93% 17.04% 26.17% 15.70% 13.97% 28.42% 24.42%
##     5:  17.24%  8.09%  4.52% 18.74% 21.06% 16.98% 25.27% 15.41% 13.48% 27.64% 23.78%
##     6:  16.21%  7.36%  4.05% 17.45% 20.60% 16.47% 24.11% 13.93% 12.65% 25.61% 22.27%
##     7:  15.17%  6.61%  3.70% 16.38% 18.77% 15.43% 22.18% 12.73% 12.07% 24.18% 21.80%
##     8:  14.17%  6.24%  3.59% 14.83% 18.38% 14.37% 20.36% 11.69% 11.13% 23.06% 19.98%
##     9:  13.32%  5.72%  3.48% 13.78% 17.18% 13.37% 19.52% 10.80% 10.45% 22.07% 18.67%
##    10:  12.25%  5.20%  3.09% 12.46% 15.67% 12.47% 17.85%  9.66%  9.51% 20.68% 17.59%
##    11:  11.51%  5.28%  2.95% 12.17% 15.25% 11.43% 16.89%  8.68%  8.57% 18.81% 16.70%
##    12:  10.71%  4.42%  2.81% 11.32% 14.32% 10.16% 15.67%  8.23%  8.57% 17.46% 15.53%
##    13:  10.00%  4.29%  2.44% 10.18% 13.55%  9.73% 14.61%  7.79%  8.05% 16.19% 14.58%
##    14:   9.47%  4.17%  2.29% 10.15% 13.28%  9.33% 13.52%  6.78%  7.42% 15.33% 13.72%
##    15:   8.97%  3.75%  2.18%  9.06% 12.56%  8.41% 13.40%  6.46%  7.60% 14.48% 12.99%
##    16:   8.59%  4.07%  2.22%  9.08% 11.93%  8.65% 12.47%  5.97%  6.80% 13.73% 12.08%
##    17:   8.23%  3.58%  2.20%  8.55% 11.93%  7.69% 11.62%  6.00%  6.88% 13.07% 11.74%
##    18:   7.85%  3.27%  2.01%  8.24% 11.06%  7.40% 11.15%  5.54%  6.70% 13.07% 11.06%
##    19:   7.54%  3.34%  1.75%  7.88% 10.78%  7.15% 10.72%  5.15%  6.54% 12.33% 10.68%
##    20:   7.26%  3.07%  1.79%  7.83% 10.34%  6.39% 10.36%  4.71%  6.36% 11.94% 10.65%
```

```
names(rf_model)
```

```
##  [1] "call"            "type"            "predicted"       "err.rate"
##  [5] "confusion"       "votes"           "oob.times"       "classes"
##  [9] "importance"      "importanceSD"    "localImportance" "proximity"
## [13] "ntree"           "mtry"            "forest"          "y"
## [17] "test"            "inbag"           "terms"
```

```
table(rf_model$predicted,train$label)
```

```
##
##        0    1    2    3    4    5    6    7    8    9
##   0 4005    2   34   25   12   37   37    7   15   20
##   1    2 4598   20   21   17   11   17   26   37   14
##   2   16   18 3849   94   16   20   29   57   76   28
##   3   10   18   48 3900    8  140    4   33  105   59
##   4    6    9   30    8 3811   23   25   47   62  126
##   5   28    7   18  126   14 3402   51    6   96   53
##   6   30    7   34   15   26   59 3942    2   35    9
##   7    6    7   50   39   20   10    2 4121   14   86
##   8   22   13   77   88   26   51   28   16 3578   51
##   9    7    3   16   34  121   42    2   86   45 3741
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.5
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
confusionMatrix(rf_model$predicted,train$label)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1    2    3    4    5    6    7    8    9
##          0 4005    2   34   25   12   37   37    7   15   20
##          1    2 4598   20   21   17   11   17   26   37   14
##          2   16   18 3849   94   16   20   29   57   76   28
##          3   10   18   48 3900    8  140    4   33  105   59
##          4    6    9   30    8 3811   23   25   47   62  126
##          5   28    7   18  126   14 3402   51    6   96   53
##          6   30    7   34   15   26   59 3942    2   35    9
##          7    6    7   50   39   20   10    2 4121   14   86
##          8   22   13   77   88   26   51   28   16 3578   51
##          9    7    3   16   34  121   42    2   86   45 3741
##
## Overall Statistics
##
##                Accuracy : 0.9274
##                  95% CI : (0.9249, 0.9299)
##     No Information Rate : 0.1115
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9194
##
##  Mcnemar's Test P-Value : 1.885e-13
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity           0.96926   0.9821  0.92170  0.89655  0.93613  0.89644
## Specificity           0.99501   0.9956  0.99064  0.98871  0.99114  0.98955
## Pos Pred Value        0.95494   0.9654  0.91577  0.90173  0.91898  0.89503
## Neg Pred Value        0.99664   0.9977  0.99135  0.98805  0.99313  0.98971
## Prevalence            0.09840   0.1115  0.09944  0.10359  0.09694  0.09037
## Detection Rate        0.09537   0.1095  0.09166  0.09287  0.09075  0.08101
## Detection Prevalence  0.09987   0.1134  0.10009  0.10299  0.09875  0.09051
## Balanced Accuracy     0.98214   0.9888  0.95617  0.94263  0.96364  0.94300
##                      Class: 6 Class: 7 Class: 8 Class: 9
## Sensitivity           0.95286  0.93638  0.88063  0.89348
```

```
## Specificity           0.99427  0.99378  0.99019  0.99058
## Pos Pred Value        0.94782  0.94627  0.90582  0.91311
## Neg Pred Value        0.99485  0.99256  0.98725  0.98823
## Prevalence            0.09851  0.10480  0.09675  0.09970
## Detection Rate        0.09387  0.09813  0.08520  0.08908
## Detection Prevalence  0.09904  0.10371  0.09406  0.09756
## Balanced Accuracy     0.97357  0.96508  0.93541  0.94203
```

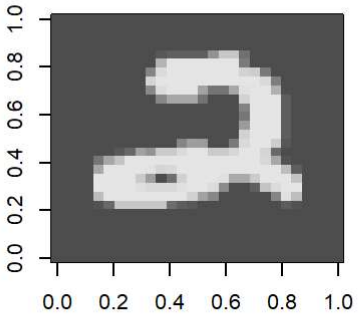
##From the above code we got that the prediction accuracy is 92%. So correctly predicted 92% and wrong prediction is 8%.

```
print("random forest trained.")
```

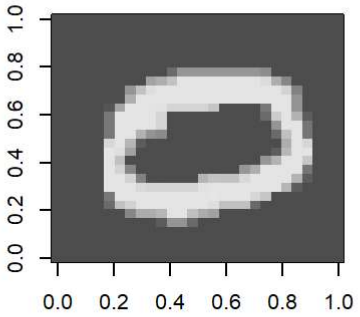
```
## [1] "random forest trained."
```

```
#prediction
pred <- predict(rf_model, newdata = test)
```

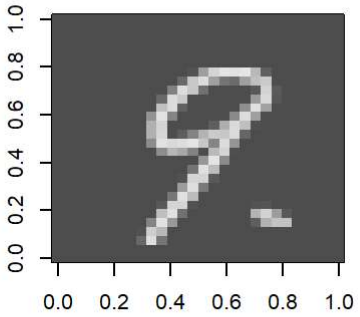
```
rotate <- function(x) t(apply(x, 2, rev)) # reverses (rotates the matrix)

# Plot a bunch of images

par(mfrow=c(2,3))
lapply(1:6,
       function(x) image(
         rotate(matrix(unlist(test[x,]),nrow = 28,byrow = T)),
         col=grey.colors(255),
         xlab=pred[x,1]
       )
)
```


0.0 0.2 0.4 0.6 0.8 1.0
0.0 0.2 0.4 0.6 0.8 1.0

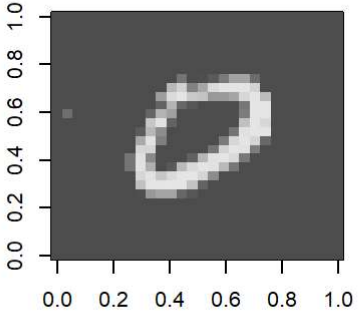
2


0.0 0.2 0.4 0.6 0.8 1.0
0.0 0.2 0.4 0.6 0.8 1.0

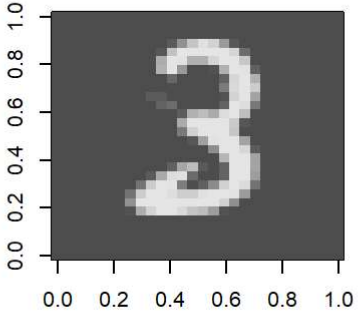
0


0.0 0.2 0.4 0.6 0.8 1.0
0.0 0.2 0.4 0.6 0.8 1.0

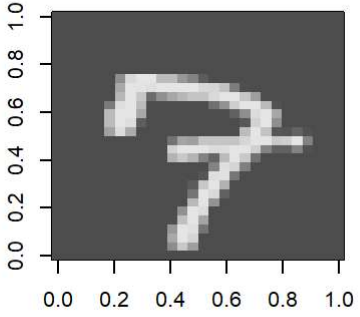
9


0.0 0.2 0.4 0.6 0.8 1.0
0.0 0.2 0.4 0.6 0.8 1.0

9


0.0 0.2 0.4 0.6 0.8 1.0
0.0 0.2 0.4 0.6 0.8 1.0

3


0.0 0.2 0.4 0.6 0.8 1.0
0.0 0.2 0.4 0.6 0.8 1.0

7

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
##
## [[5]]
## NULL
##
## [[6]]
## NULL
```

```
par(mfrow=c(1,1)) # set plot options back to default
```

conclusion

As per the conclusion of the above problem , we got 92% accuracy in the trainig model .But in our test dataset doesnot contain response variable so couldnot make the accuracy. So we can conclude taht Random forest algorithm is giving better perfomance ,then we can use random forest algorithm for this kind of problems.