Handwritten Digit Recognition

NIANG Mohamed
KAINA Mohamed Abdellah
NGREMMADJI Mbaimou Auxence

31 Octobre 2019

Contents

1 Preliminaries **1**
- 1.1 Set Working Directory . . . 1
- 1.2 Load libraries/data/create new variables . . . 2

2 Data Exploratory Analysis **3**
- 2.1 Look at the TRAINING data set . . . 3
- 2.2 Plot the value of the first four examples of the zip.train . . . 3
- 2.3 Look at the TEST data set . . . 4
- 2.4 Plot the value of the first four examples of the zip.test . . . 4
- 2.5 Chose the two digits 0 and 8 in the Training Set . . . 5
- 2.6 Chose the two digits 0 and 8 in the Test Set . . . 5
- 2.7 Find number of missing values/check ranges (TRAINING data set) . . . 5
- 2.8 Find number of missing values/check ranges (TESTING data set) . . . 5
- 2.9 Transformation. Transform Label as Factor (Categorical) and Change Column Names (TRAINING data set) . . . 5
- 2.10 Transformation. Transform Label as Factor (Categorical) and Change Column Names (TESTING data set) . . . 6
- 2.11 Total Number of Digits (Training Set) . . . 6
- 2.12 Total Number of Digits (Testing Set) . . . 7

3 Machine Learning Classifiers **7**
- 3.1 Classification. Predictive Model. Naive Bayes Algorithm . . . 7
- 3.2 Classification. k-Nearest Neighbors (kNN) Algorithm . . . 8
- 3.3 Classification. Fast Nearest Neighbors (FNN) Algorithm . . . 9
- 3.4 Classification. Predictive Model. SVM (Support Vector Machine) Algorithm . . . 9
- 3.5 Classification. Predictive Model. RPart (Recursive Partitioning and Regression Trees) Algorithm 10
- 3.6 Classification. Predictive Model. Bagging Algorithm . . . 12
- 3.7 Classification. Predictive Model. Random Forest Algorithm . . . 13

4 Model comparison and Conclusion **13**
- 4.1 Model Comparison . . . 13
- 4.2 Conclusion . . . 14

This is a classification problem (Machine Learning)

The two digits we have chosen for the classification are 0 and 8

1 Preliminaries

1.1 Set Working Directory

```
WORKING_DIR <- "C:/Users/HP/Desktop/ML PROJECT"
COLORS <- c('white','black')
```

```r
setwd(WORKING_DIR)
```

1.2 Load libraries/data/create new variables

```r
# Load Libraries
library(MASS)
library(plyr)
library(knitr) # Markdown
library(RColorBrewer)
library(ElemStatLearn)
```

```
##
## Attaching package: 'ElemStatLearn'

## The following object is masked from 'package:plyr':
##
##     ozone
```

```r
library(foreign) # For reading and writing data stored
library(mlbench)
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```r
library(tree)
library(maptree)
```

```
## Loading required package: cluster

## Loading required package: rpart
```

```r
library(rpart.plot)
library(rpart) # Recursive Partitioning and Regression Trees (RPart)
library(ipred)
library(randomForest)
```

```
## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(RWeka) # Weka
```

```
##
## Attaching package: 'RWeka'

## The following objects are masked from 'package:foreign':
##
##     read.arff, write.arff
```

```r
library(FNN) # Fast k-Nearest Neighbors (kNN)
library(e1071) # Support Vector Machine (SVM)

# Set Color
# colorRampPalette(COLORS) ( 4 )
```

```
CUSTOM_COLORS <- colorRampPalette(colors=COLORS)
CUSTOM_COLORS_PLOT <- colorRampPalette(brewer.pal(10,"Set3"))

# Load data
data(zip.train)
data(zip.test)

DATASET.train <- as.data.frame(zip.train)
DATASET.test <- as.data.frame(zip.test)
```

---

2 Data Exploratory Analysis

2.1 Look at the TRAINING data set

2.1.1 Check the number of observations in the zip.train (ntrain = 7291)

```
dim(DATASET.train)
```

```
## [1] 7291  257
```

2.2 Plot the value of the first four examples of the zip.train

```
par(mfrow=c(1,4));
image(zip2image(zip.train,1), col=gray(256:0/256), zlim=c(0,1), xlab="", ylab="",xaxt="n",yaxt="n")
```

```
## [1] "digit  6  taken"
```

```
image(zip2image(zip.train,2), col=gray(256:0/256), zlim=c(0,1), xlab="", ylab="",xaxt="n",yaxt="n")
```

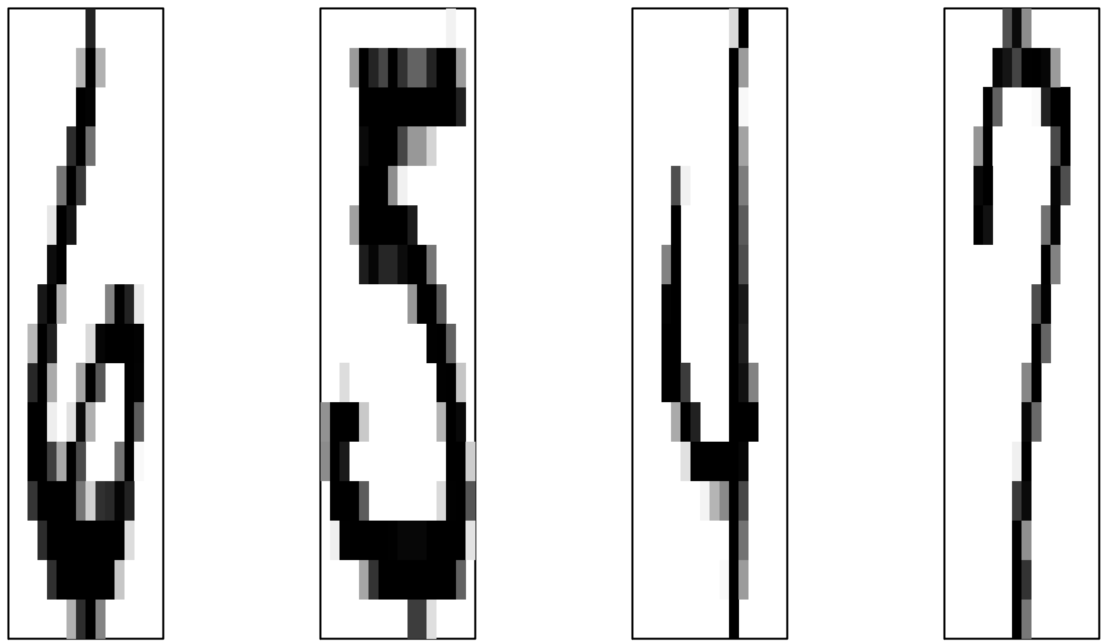
```
## [1] "digit  5  taken"
```

```
image(zip2image(zip.train,3), col=gray(256:0/256), zlim=c(0,1), xlab="", ylab="",xaxt="n",yaxt="n")
```

```
## [1] "digit  4  taken"
```

```
image(zip2image(zip.train,4), col=gray(256:0/256), zlim=c(0,1), xlab="", ylab="",xaxt="n",yaxt="n")
```

```
## [1] "digit  7  taken"
```



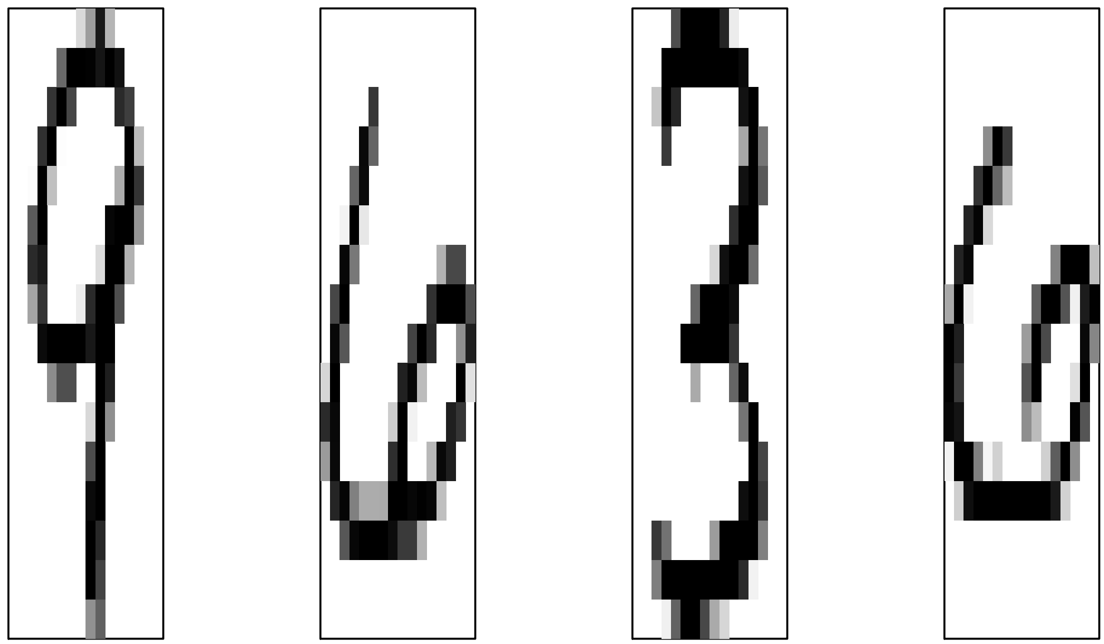
2.3 Look at the TEST data set

2.3.1 Check the number of observations in the zip.test (ntest = 2007)

```
dim(DATASET.test)

## [1] 2007  257
```

2.4 Plot the value of the first four examples of the zip.test

```
par(mfrow=c(1,4));
image(zip2image(zip.test,1), col=gray(256:0/256), zlim=c(0,1), xlab="", ylab="",xaxt="n",yaxt="n")

## [1] "digit  9  taken"

image(zip2image(zip.test,2), col=gray(256:0/256), zlim=c(0,1), xlab="", ylab="",xaxt="n",yaxt="n")

## [1] "digit  6  taken"

image(zip2image(zip.test,3), col=gray(256:0/256), zlim=c(0,1), xlab="", ylab="",xaxt="n",yaxt="n")

## [1] "digit  3  taken"

image(zip2image(zip.test,4), col=gray(256:0/256), zlim=c(0,1), xlab="", ylab="",xaxt="n",yaxt="n")

## [1] "digit  6  taken"
```



2.5 Chose the two digits 0 and 8 in the Training Set

```
DATASET.train <- DATASET.train[ which(DATASET.train$V1 == '0' | DATASET.train$V1 == '8'), ]
```

2.6 Chose the two digits 0 and 8 in the Test Set

```
DATASET.test <- DATASET.test[ which(DATASET.test$V1 == '0' | DATASET.test$V1 == '8'), ]
```

2.7 Find number of missing values/check ranges (TRAINING data set)

```
sum(is.na(DATASET.train))
```

```
## [1] 0
```

2.8 Find number of missing values/check ranges (TESTING data set)

```
sum(is.na(DATASET.test))
```

```
## [1] 0
```

2.9 Transformation. Transform Label as Factor (Categorical) and Change Column Names (TRAINING data set)

```
DATASET.train[,1] <- as.factor(DATASET.train[,1]) # As Category
colnames(DATASET.train) <- c("Y",paste("X.",1:256,sep=""))
class(DATASET.train[,1])
```

```
## [1] "factor"
```

```
levels(DATASET.train[,1])
```

```
## [1] "0" "8"
```

2.10 Transformation. Transform Label as Factor (Categorical) and Change Column Names (TESTING data set)

```
DATASET.test[,1] <- as.factor(DATASET.test[,1]) # As Category
colnames(DATASET.test) <- c("Y",paste("X.",1:256,sep=""))
class(DATASET.test[,1])
```

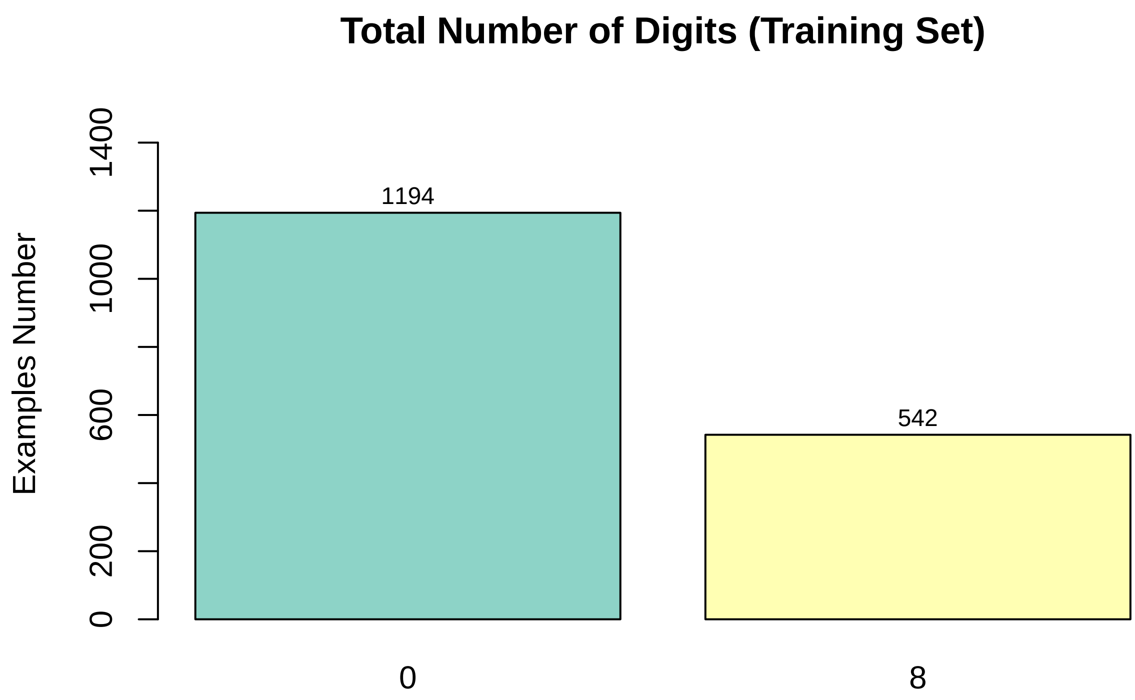
```
## [1] "factor"
```

```
levels(DATASET.test[,1])
```

```
## [1] "0" "8"
```

2.11 Total Number of Digits (Training Set)

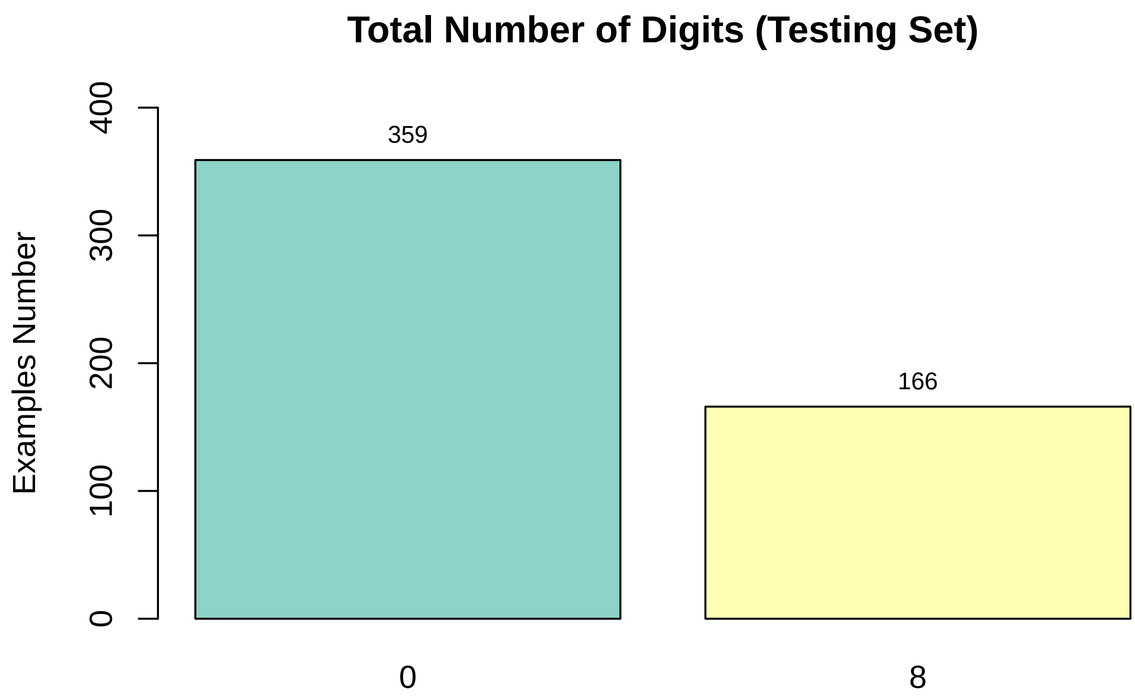
```
resTable <- table(DATASET.train$Y)
par(mfrow=c(1,1))
par(mar=c(5, 4, 4, 2) + 0.1) # increase y-axis margin.
plot <- plot(DATASET.train$Y,col=CUSTOM_COLORS_PLOT(10), main="Total Number of Digits (Training Set)", y
text(x=plot,y=resTable+50, labels=resTable, cex=0.75)
```




```
par(mfrow=c(1,1))
percentage <- round(resTable/sum(resTable)*100)
labels <- paste0 (row.names(resTable), " (", percentage ,"%) ") # add percents to labels
```

2.12 Total Number of Digits (Testing Set)

```
resTable <- table(DATASET.test$Y)
par(mfrow=c(1,1))
par(mar=c(5, 4, 4, 2) + 0.1) # increase y-axis margin.
plot <- plot(DATASET.test$Y,col=CUSTOM_COLORS_PLOT(10), main="Total Number of Digits (Testing Set)", yli
text(x=plot,y=resTable+20, labels=resTable, cex=0.75)
```




```
par(mfrow=c(1,1))
percentage <- round(resTable/sum(resTable)*100)
labels <- paste0 (row.names(resTable), " (", percentage ,"%) ") # add percents to labels
```

---

3 Machine Learning Classifiers

3.1 Classification. Predictive Model. Naive Bayes Algorithm

```
pc <- proc.time()
model.naiveBayes <- naiveBayes(DATASET.train$Y ~ . , data=DATASET.train)
proc.time() - pc
```

```
##    user  system elapsed
##    0.09    0.00    0.09
```

```
summary(model.naiveBayes)
```

```
##           Length Class  Mode
## apriori    2     table  numeric
```

```
## tables    256    -none- list
## levels      2    -none- character
## isnumeric 256    -none- logical
## call        4    -none- call
```

3.1.1 Confusion Matrix (naiveBayes)

```
prediction.naiveBayes <- predict(model.naiveBayes, newdata=DATASET.test, type='class')
table("Actual Class"=DATASET.test$Y, "Predicted Class"=prediction.naiveBayes)
```

```
##               Predicted Class
## Actual Class   0   8
##              0 319  40
##              8   4 162
```

```
error.rate.naiveBayes <- sum(DATASET.test$Y != prediction.naiveBayes) / nrow(DATASET.test)
accuracy.naiveBayes <- 1 - error.rate.naiveBayes
print (paste0("Accuary (Precision): ", accuracy.naiveBayes))
```

```
## [1] "Accuary (Precision): 0.916190476190476"
```

3.2 Classification. k-Nearest Neighbors (kNN) Algorithm

```
pc <- proc.time()
model.knn <- IBk(DATASET.train$Y ~ . , data=DATASET.train)
proc.time() - pc
```

```
##    user  system elapsed
##    2.14    0.04    1.83
```

```
summary(model.knn)
```

```
##
## === Summary ===
##
## Correctly Classified Instances        1736               100      %
## Incorrectly Classified Instances         0                 0      %
## Kappa statistic                          1
## Mean absolute error                      0.0006
## Root mean squared error                  0.0006
## Relative absolute error                  0.1339 %
## Root relative squared error              0.1242 %
## Total Number of Instances             1736
##
## === Confusion Matrix ===
##
##     a    b   <-- classified as
##  1194    0 |    a = 0
##     0  542 |    b = 8
```

3.2.1 Confusion Matrix (kNN)

```
prediction.knn <- predict(model.knn, newdata=DATASET.test, type='class')
table("Actual Class"=DATASET.test$Y, "Predicted Class"=prediction.knn)
```

```
##               Predicted Class
## Actual Class   0   8
```

```
##           0 356   3
##           8   6 160
```

```
error.rate.knn <- sum(DATASET.test$Y != prediction.knn) / nrow(DATASET.test)
accuracy.knn <- 1 - error.rate.knn
print (paste0("Accuary (Precision): ", accuracy.knn))
```

```
## [1] "Accuary (Precision): 0.982857142857143"
```

3.3 Classification. Fast Nearest Neighbors (FNN) Algorithm

```
pc <- proc.time()
# Avoid Name Collision (knn)
model.fnn <- FNN::knn(DATASET.train[,-1], DATASET.test[, -1], DATASET.train$Y, k = 10, algorithm="cover
proc.time() - pc
```

```
##    user  system elapsed
##    0.50    0.00    0.51
```

```
summary(model.fnn)
```

```
##   0   8
## 367 158
```

3.3.1 Confusion Matrix (FNN)

```
table("Actual Class"=DATASET.test$Y, "Predicted Class"=model.fnn)
```

```
##             Predicted Class
## Actual Class   0   8
##            0 358   1
##            8   9 157
```

```
error.rate.fnn <- sum(DATASET.test$Y != model.fnn) / nrow(DATASET.test)
accuracy.fnn <- 1 - error.rate.fnn
print (paste0("Accuary (Precision): ", accuracy.fnn))
```

```
## [1] "Accuary (Precision): 0.980952380952381"
```

3.4 Classification. Predictive Model. SVM (Support Vector Machine) Algorithm

```
pc <- proc.time()
model.svm <- svm(DATASET.train$Y ~ . ,method="class", data=DATASET.train)
```

```
## Warning in svm.default(x, y, scale = scale, ..., na.action = na.action):
## Variable(s) 'X.256' constant. Cannot scale data.
```

```
proc.time() - pc
```

```
##    user  system elapsed
##    0.61    0.02    0.64
```

```
summary(model.svm)
```

```
##
## Call:
## svm(formula = DATASET.train$Y ~ ., data = DATASET.train, method = "class")
```

```
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  153
##
##  ( 76 77 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 8
```

3.4.1 Confusion Matrix (SVM)

```
prediction.svm <- predict(model.svm, newdata=DATASET.test, type='class')
table("Actual Class"=DATASET.test$Y, "Predicted Class"=prediction.svm)
```

```
##             Predicted Class
## Actual Class   0   8
##            0 357   2
##            8   4 162
```

```
error.rate.svm <- sum(DATASET.test$Y != prediction.svm) / nrow(DATASET.test)
accuracy.svm <- 1 - error.rate.svm
print (paste0("Accuary (Precision): ", accuracy.svm))
```

```
## [1] "Accuary (Precision): 0.988571428571429"
```

3.5 Classification. Predictive Model. RPart (Recursive Partitioning and Regression Trees) Algorithm

```
pc <- proc.time()
model.rpart <- rpart(DATASET.train$Y ~ . ,method="class", data=DATASET.train)
proc.time() - pc
```

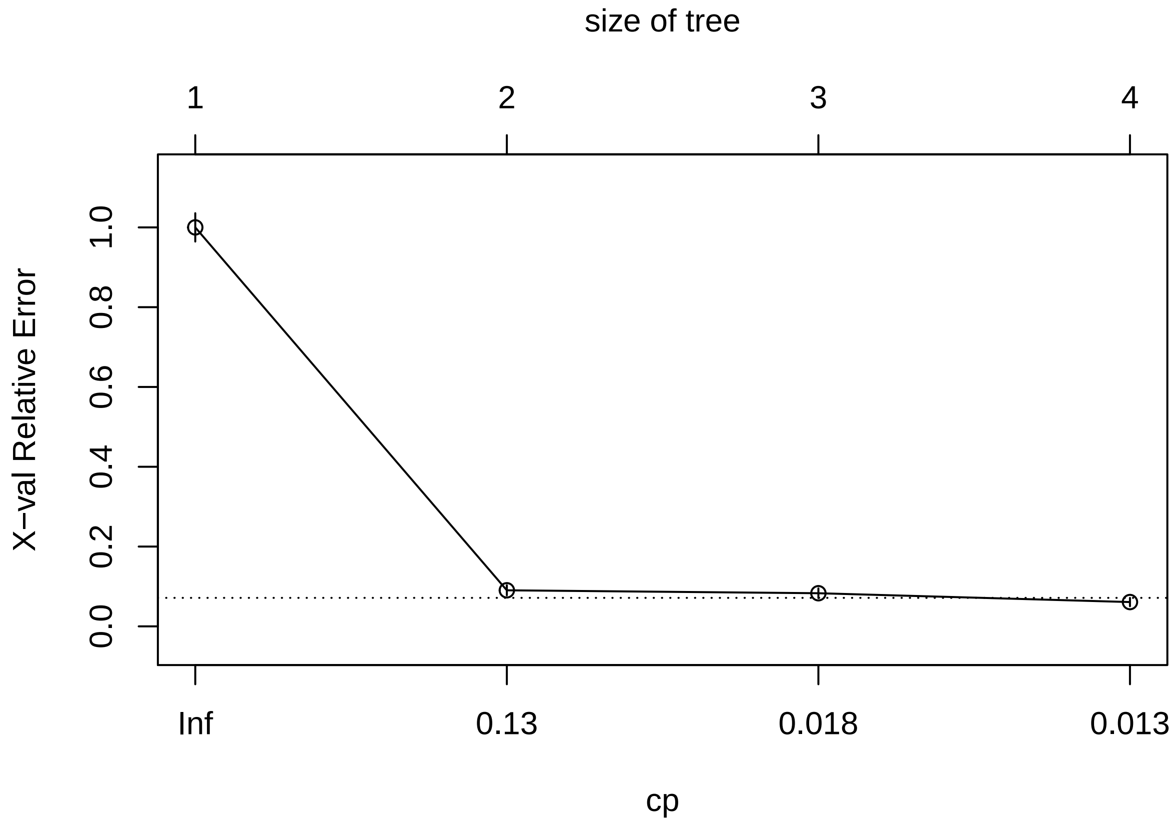
```
##    user  system elapsed
##    0.57    0.00    0.56
```

```
printcp(model.rpart)
```

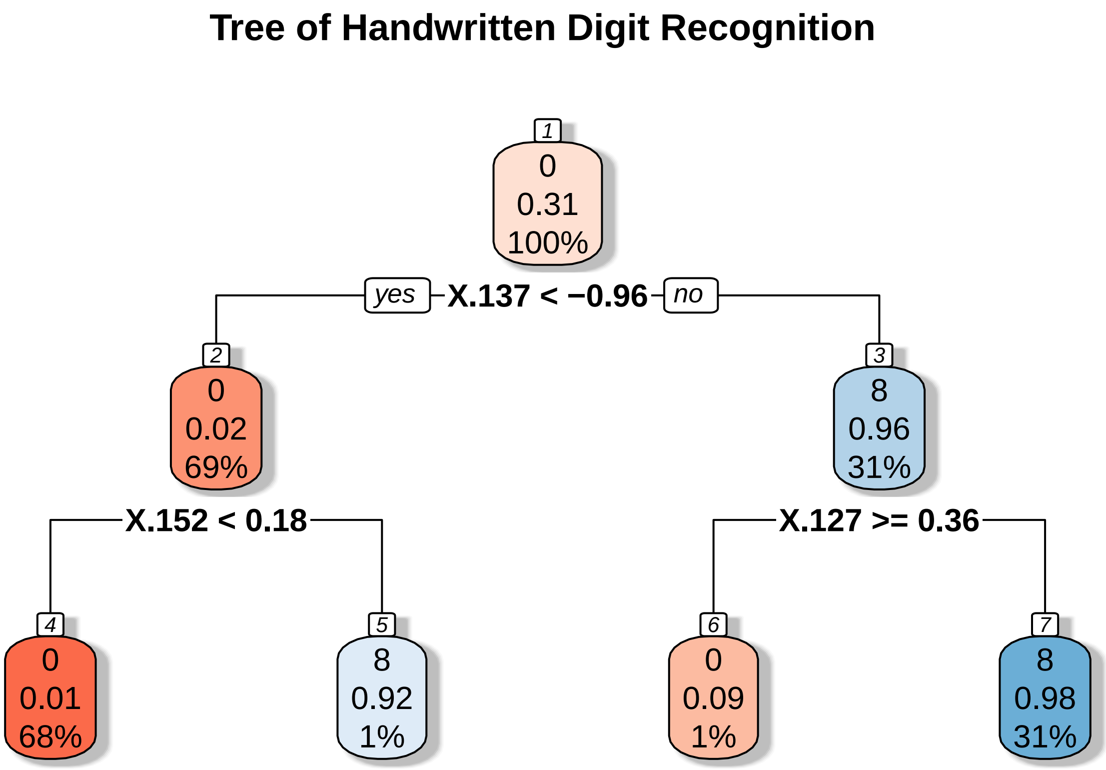
```
##
## Classification tree:
## rpart(formula = DATASET.train$Y ~ ., data = DATASET.train, method = "class")
##
## Variables actually used in tree construction:
## [1] X.127 X.137 X.152
##
## Root node error: 542/1736 = 0.31221
##
## n= 1736
##
```

```
##         CP nsplit rel error   xerror     xstd
## 1 0.916974      0  1.000000 1.000000 0.035623
## 2 0.018450      1  0.083026 0.090406 0.012732
## 3 0.016605      2  0.064576 0.083026 0.012215
## 4 0.010000      3  0.047970 0.060886 0.010498
```

```r
plotcp(model.rpart) # visualize cross-validation results
```




```r
rpart.plot(model.rpart, box.palette="RdBu", shadow.col="gray", nn=TRUE, uniform=TRUE, main="Tree of Han
```



3.5.1 Confusion Matrix (RPart)

```
prediction.rpart <- predict(model.rpart, newdata=DATASET.test, type='class')
table("Actual Class"=DATASET.test$Y, "Predicted Class"=prediction.rpart)
```

```
##              Predicted Class
## Actual Class   0   8
##             0 347  12
##             8   7 159
```

```
error.rate.rpart <- sum(DATASET.test$Y != prediction.rpart) / nrow(DATASET.test)
accuracy.rpart <- 1 - error.rate.rpart
print (paste0("Accuary (Precision): ", accuracy.rpart))
```

```
## [1] "Accuary (Precision): 0.963809523809524"
```

3.6 Classification. Predictive Model. Bagging Algorithm

```
pc <- proc.time()
model.bagging <- bagging(DATASET.train$Y ~ . , method="class", data=DATASET.train, coob = TRUE, control
proc.time() - pc
```

```
##    user  system elapsed
##   15.33    0.09   15.44
```

3.6.1 Confusion Matrix (Bagging)

```
prediction.bagging <- predict(model.bagging, newdata=DATASET.test, type='class')
table("Actual Class"=DATASET.test$Y, "Predicted Class"=prediction.bagging)
```

```
##              Predicted Class
## Actual Class   0   8
##             0 353   6
##             8   6 160
```

```
error.rate.bagging <- sum(DATASET.test$Y != prediction.bagging) / nrow(DATASET.test)
accuracy.bagging <- 1 - error.rate.bagging
print (paste0("Accuary (Precision): ", accuracy.bagging))
```

```
## [1] "Accuary (Precision): 0.977142857142857"
```

3.7 Classification. Predictive Model. Random Forest Algorithm

```
pc <- proc.time()
model.forest <- randomForest(DATASET.train$Y ~ . , method="class", data=DATASET.train, importance=TRUE)
proc.time() - pc
```

```
##    user  system elapsed
##   10.75    0.03   10.80
```

3.7.1 Confusion Matrix (Random Forest)

```
prediction.forest <- predict(model.forest, newdata=DATASET.test, type='class')
table("Actual Class"=DATASET.test$Y, "Predicted Class"=prediction.forest)
```

```
##              Predicted Class
## Actual Class   0   8
##             0 357   2
##             8   4 162
```

```
error.rate.forest <- sum(DATASET.test$Y != prediction.forest) / nrow(DATASET.test)
accuracy.forest <- 1 - error.rate.forest
print (paste0("Accuary (Precision): ", accuracy.forest))
```

```
## [1] "Accuary (Precision): 0.988571428571429"
```

4 Model comparison and Conclusion

4.1 Model Comparison

```
models <- c('naiveBayes', 'knn', 'fnn', 'svm', 'rpart', 'bagging', 'randomforest')
accuracy <- c(accuracy.naiveBayes, accuracy.knn, accuracy.fnn, accuracy.svm, accuracy.rpart, accuracy.ba
results <- data.frame("Models" = models, "Accuracy" = accuracy)
# Table comparison
kable(arrange(results,desc(accuracy)))
```

Models	Accuracy
svm	0.9885714
randomforest	0.9885714
knn	0.9828571
fnn	0.9809524
bagging	0.9771429
rpart	0.9638095

Models	Accuracy
naiveBayes	0.9161905

4.2 Conclusion

From the results of the different models we have had, it seems that **svm** gives better results and could be used for prediction for new observations.