HW-6-Subbu-Kandhaswamy-IST707

Subbu Kandhaswamy

11/14/2019

## HomeWork-6

Naïve Bayes and decision tree for handwriting recognition. The goal is to recognize digits 0 to 9 in handwriting images. We are going to use the sampled data to construct prediction models using naïve Bayes and decision tree algorithms. Tune their parameters to get the best model (measured by cross validation) and compare which algorithms provide better model for this task.

#Load required libraries  
  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(rpart.plot)

## Loading required package: rpart

library(ggplot2)  
library(e1071)  
  
#Load train/test data  
  
kg\_train\_data<-read.csv("C:/Users/Administrator/Documents/MSDatascience/IST707/Week6/train.csv")  
kg\_test\_data<-read.csv("C:/Users/Administrator/Documents/MSDatascience/IST707/Week6/test.csv")  
  
#delete redundant pixels  
delete = c(  
 'pixel0',  
 'pixel1',  
 'pixel2',  
 'pixel3',  
 'pixel4',  
 'pixel5',  
 'pixel6',  
 'pixel7',  
 'pixel8',  
 'pixel9',  
 'pixel10',  
 'pixel11',  
 'pixel780',  
 'pixel781',  
 'pixel782',  
 'pixel783'  
)  
  
kg\_train\_data = kg\_train\_data[, !(names(kg\_train\_data) %in% delete)]  
kg\_test\_data = kg\_test\_data[, !(names(kg\_test\_data) %in% delete)]  
  
dim(kg\_train\_data)

## [1] 42000 769

dim(kg\_test\_data)

## [1] 28000 768

kg\_train\_data$label <- as.factor(kg\_train\_data$label)  
#Split  
  
#Splitting dataset into 10% and use it for classifiers  
trainSplit <- sample(nrow(kg\_train\_data), nrow(kg\_train\_data)\* .1)  
testSplit<- sample(nrow(kg\_test\_data), nrow(kg\_test\_data)\* .1)  
  
trainSubset <- kg\_train\_data[trainSplit,]  
testSubset <- kg\_test\_data[testSplit,]  
  
#Decision Tree Model  
  
decTreeTrain <- rpart(label ~ ., data = trainSubset, method = 'class', control= rpart.control(cp = 0), minsplit = 100, maxdepth = 10)  
  
#Next Testing accuracy of the model on Training Data  
  
trainPred <- data.frame(predict(decTreeTrain, trainSubset))  
  
trainPred <- as.data.frame(names(trainPred[apply(trainPred,1,which.max)]))  
colnames(trainPred) <- 'prediction'  
  
trainPred$number <- substr(trainPred$prediction, 2,2)  
  
trainPred <- trainSubset %>% bind\_cols(trainPred) %>% select(label, number) %>% mutate(label=as.factor(label), number = as.factor(number))  
  
  
#Confusion Matrix  
confusionMatrix(trainPred$label, trainPred$number)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1 2 3 4 5 6 7 8 9  
## 0 392 0 1 5 1 1 10 3 3 6  
## 1 0 426 0 2 3 1 3 5 5 0  
## 2 13 7 300 13 2 3 9 10 15 11  
## 3 1 7 14 385 2 16 0 6 11 8  
## 4 4 2 5 3 351 4 11 7 6 27  
## 5 6 8 10 17 15 315 10 5 16 4  
## 6 3 8 11 3 10 12 341 4 2 4  
## 7 0 5 6 7 5 4 0 401 8 5  
## 8 0 9 1 20 9 8 7 2 367 9  
## 9 1 3 6 9 22 3 5 9 12 333  
##   
## Overall Statistics  
##   
## Accuracy : 0.8598   
## 95% CI : (0.8489, 0.8701)  
## No Information Rate : 0.1131   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8441   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.93333 0.8968 0.84746 0.82974 0.83571 0.85831  
## Specificity 0.99206 0.9949 0.97842 0.98260 0.98175 0.97626  
## Pos Pred Value 0.92891 0.9573 0.78329 0.85556 0.83571 0.77586  
## Neg Pred Value 0.99259 0.9870 0.98585 0.97893 0.98175 0.98629  
## Prevalence 0.10000 0.1131 0.08429 0.11048 0.10000 0.08738  
## Detection Rate 0.09333 0.1014 0.07143 0.09167 0.08357 0.07500  
## Detection Prevalence 0.10048 0.1060 0.09119 0.10714 0.10000 0.09667  
## Balanced Accuracy 0.96270 0.9459 0.91294 0.90617 0.90873 0.91728  
## Class: 6 Class: 7 Class: 8 Class: 9  
## Sensitivity 0.86111 0.88717 0.82472 0.81818  
## Specificity 0.98502 0.98933 0.98269 0.98154  
## Pos Pred Value 0.85678 0.90930 0.84954 0.82630  
## Neg Pred Value 0.98553 0.98643 0.97930 0.98051  
## Prevalence 0.09429 0.10762 0.10595 0.09690  
## Detection Rate 0.08119 0.09548 0.08738 0.07929  
## Detection Prevalence 0.09476 0.10500 0.10286 0.09595  
## Balanced Accuracy 0.92306 0.93825 0.90370 0.89986

# Naive Bayes  
  
nbTrain <- naiveBayes(as.factor(label) ~ ., data = trainSubset)  
  
nbTrainPred <- predict(nbTrain, trainSubset, type = 'class')   
  
confusionMatrix(nbTrainPred, as.factor(trainSubset$label))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1 2 3 4 5 6 7 8 9  
## 0 391 0 38 60 6 56 10 5 9 1  
## 1 2 433 26 38 18 23 16 24 90 19  
## 2 1 0 86 4 0 1 0 0 0 0  
## 3 0 0 22 107 0 1 0 2 3 0  
## 4 2 0 4 15 131 6 2 16 4 9  
## 5 0 0 34 41 4 202 8 1 9 4  
## 6 14 7 113 34 32 26 348 3 13 3  
## 7 0 0 4 1 0 0 0 98 0 1  
## 8 3 3 42 95 5 49 5 5 196 6  
## 9 9 2 14 55 224 42 9 287 108 360  
##   
## Overall Statistics  
##   
## Accuracy : 0.56   
## 95% CI : (0.5448, 0.5751)  
## No Information Rate : 0.1071   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5114   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.9265 0.9730 0.22454 0.23778 0.31190 0.49754  
## Specificity 0.9510 0.9318 0.99843 0.99253 0.98466 0.97338  
## Pos Pred Value 0.6788 0.6284 0.93478 0.79259 0.69312 0.66667  
## Neg Pred Value 0.9914 0.9966 0.92770 0.91562 0.92795 0.94765  
## Prevalence 0.1005 0.1060 0.09119 0.10714 0.10000 0.09667  
## Detection Rate 0.0931 0.1031 0.02048 0.02548 0.03119 0.04810  
## Detection Prevalence 0.1371 0.1640 0.02190 0.03214 0.04500 0.07214  
## Balanced Accuracy 0.9388 0.9524 0.61149 0.61516 0.64828 0.73546  
## Class: 6 Class: 7 Class: 8 Class: 9  
## Sensitivity 0.87437 0.22222 0.45370 0.89330  
## Specificity 0.93556 0.99840 0.94347 0.80248  
## Pos Pred Value 0.58685 0.94231 0.47922 0.32432  
## Neg Pred Value 0.98614 0.91626 0.93775 0.98608  
## Prevalence 0.09476 0.10500 0.10286 0.09595  
## Detection Rate 0.08286 0.02333 0.04667 0.08571  
## Detection Prevalence 0.14119 0.02476 0.09738 0.26429  
## Balanced Accuracy 0.90497 0.61031 0.69859 0.84789

## Comparison of the Algorithms Performance

According to our Algorithmic models, Decision tree performs better than Naïve Bayes for this set of data.

Decision Tree’s predicting accuracy is 85.98% , while Naïve Bayes has a predicting ability of 56% which is quite low. Naïve Bayes had problem determining 9 which can be fixed if the dataset was large. However, computations were performed on random sample of 10% data, but if we choose anther random sample or large sample data results could have been better.

Decision trees are very flexible, easy to understand, and easy to debug. They will work with classification problems and regression problems. Naive Bayes requires you build a classification by hand. There’s no way to just toss a bunch of tabular data at it and have it pick the best features it will use to classify. As such there’s no better classifier, it depends upon problem to problem.

Naive Bayes:

1. Work well with small dataset compared to DT which need more data

2. Lesser overfitting

3. Smaller in size and faster in processing

Decision Tree:

1. Decision Trees are very flexible, easy to understand, and easy to debug

2. No preprocessing or transformation of features required

3. Prone to overfitting but you can user pruning or Random forests to avoid that.