Homework 6 - Alex Hyman

Alex Hyman

11/18/2018

## Introduction

Using the pixel intensity of hand drawn digits (0-9) can a supervised machine learning algorithm correctly classift which number is drawn? The data provided contains the pixel intensity for 42,000 different 28x28 images. Each of the images are a digit, 0-9 and each of the values provided are the pixel intensities of a specific pixel in the given image.

The generic preprocessing for this problem involved separating the labeled dataset into a training and test set. The training set contained 80% of the data and the test set contained the remaining 20% of the labeled data. The labels also needed to be converted into factors because this is not a regression problem, but a classification problem. The final training set contained 33,604 hand-drawn digits, while the test set contained 8,396 different hand-drawn digits.

library(dplyr)

library(tidyr)  
library(caret)

library(klaR)

library(rpart)  
library(rpart.plot)  
library(matrixStats)

library(arules)

library(plyr)

#Reading the training file  
data <- read.csv("train.csv")  
#seeing if any missing data  
sum(is.na(data))

## [1] 0

#converting the label to a factor  
data$label <- as.factor(data$label)  
#Reading the data that will be prdicted  
validation <- read.csv("test.csv")  
#getting dimensions of data  
dim(data)

## [1] 42000 785

set.seed(1000)  
#Getting the index for the training data  
train\_ix <- createDataPartition(data$label, p = 0.8, list = F)  
#Training data frame  
train\_df <- data[train\_ix, ]  
#Test data set  
test\_df <- data[-train\_ix, ]  
  
#getting new size of the train index  
dim(train\_df)

## [1] 33604 785

dim(test\_df)

## [1] 8396 785

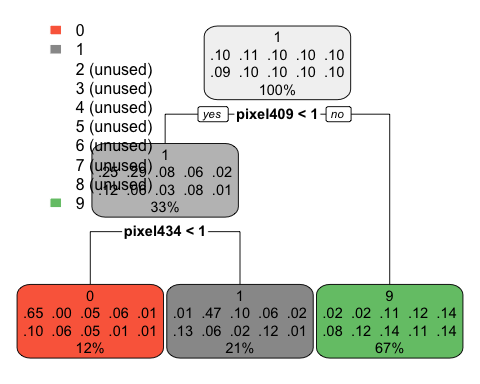
## Decision Tree

An initial decision tree model was created to classify the different hand-drawn digits. A 3-fold cross-validation was conducted on the training set and a cross-validation accuracy of 0.31 was found for the initial model. A plot of the decision tree model shows that the model only had three different leaf nodes, for digits 0, 1, and 9. This would only allow the model to correctly identify 30% of all the digits in the training set.

#Making the training parameters  
tc <- trainControl(method = "cv", number = 3)  
#intial decision tree model  
dt <- train(label ~ ., data = train\_df, trControl = tc, method = "rpart")  
#printing the decision tree  
print(dt)

## CART   
##   
## 33604 samples  
## 784 predictor  
## 10 classes: '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'   
##   
## No pre-processing  
## Resampling: Cross-Validated (3 fold)   
## Summary of sample sizes: 22402, 22403, 22403   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.07358655 0.3161541 0.23681696  
## 0.09083601 0.2197044 0.12996575  
## 0.09177385 0.1381094 0.03260562  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.07358655.

#printing the decision tree  
rpart.plot(dt$finalModel)



#decreasing the complexity

The confusion matrix for the decision tree model shows the issue with only having three leaf nodes, While the performance is decent on the digits 0, 1, and 9, everything else is misclassified as it the model does not have any branches that take digits 2-8 into account. A newly tuned decision tree should be made that allows for more branches.

#Predicting the   
preds <- predict(dt, test\_df, "raw")  
#Confusion matrix  
confusionMatrix(preds, test\_df$label)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1 2 3 4 5 6 7 8 9  
## 0 674 2 53 60 8 105 59 63 9 14  
## 1 12 833 180 113 33 211 93 38 203 19  
## 2 0 0 0 0 0 0 0 0 0 0  
## 3 0 0 0 0 0 0 0 0 0 0  
## 4 0 0 0 0 0 0 0 0 0 0  
## 5 0 0 0 0 0 0 0 0 0 0  
## 6 0 0 0 0 0 0 0 0 0 0  
## 7 0 0 0 0 0 0 0 0 0 0  
## 8 0 0 0 0 0 0 0 0 0 0  
## 9 140 101 602 697 773 443 675 779 600 804  
##   
## Overall Statistics  
##   
## Accuracy : 0.2753   
## 95% CI : (0.2657, 0.2849)  
## No Information Rate : 0.1115   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.193   
## Mcnemar's Test P-Value : NA

Multiple complexity parameter were attempted to find a better decision tree model to predict which class each of the hand-written digits belonged to, and the best parameter found was 0.0005. This value resulted in a cross-validation accuracy of 0.82, and an accuracy on the test set of 0.82. The similarity between the cross-validation accuracy and the test accuracy shows that the model was not overfit, and the model performed fairly well. There were not any pre-processing steps that were required when building the decision tree model, as the decision tree dies not expect any distributions. This decision tree model was too big to print.

#Adjusting the complexity parameter  
dt\_grid <- expand.grid("cp" = 0.0005)  
#New decision tree   
dt\_tuned <- train(label ~ ., data = train\_df, trControl = tc,   
 tuneGrid = dt\_grid, method = "rpart")  
#cross validation accuracy  
dt\_tuned

## CART   
##   
## 33604 samples  
## 784 predictor  
## 10 classes: '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'   
##   
## No pre-processing  
## Resampling: Cross-Validated (3 fold)   
## Summary of sample sizes: 22402, 22403, 22403   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.817224 0.7968291  
##   
## Tuning parameter 'cp' was held constant at a value of 5e-04

#PRedictinf with newly tuned decision tree  
preds\_tuned <- predict(dt\_tuned$finalModel, test\_df, type = "class")  
#Confusion matrix of new decision tree  
confusionMatrix(preds\_tuned, test\_df$label)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1 2 3 4 5 6 7 8 9  
## 0 752 1 26 4 6 23 24 22 5 5  
## 1 2 880 22 12 17 4 8 10 27 5  
## 2 13 11 677 45 12 12 28 28 30 9  
## 3 7 11 22 635 8 46 3 2 33 23  
## 4 6 4 19 12 659 18 20 12 11 40  
## 5 14 4 7 48 5 547 17 8 22 12  
## 6 5 3 18 15 19 32 688 0 15 3  
## 7 9 3 20 31 4 18 3 747 4 12  
## 8 13 9 10 46 22 27 20 7 627 14  
## 9 5 10 14 22 62 32 16 44 38 714  
##   
## Overall Statistics  
##   
## Accuracy : 0.8249   
## 95% CI : (0.8166, 0.833)  
## No Information Rate : 0.1115   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8054   
## Mcnemar's Test P-Value : < 2.2e-16   
##

## Naive Bayes

A naive bayes model was also created to classify the hand-written digits. The preprocessing involved in creating this model involved looking at each of the specific pixels, and only allowing pixels that have 10,000 or more instances of a marking in that pixel. The columns were also discretized so that the data objects were binary, where if the pizel value was greater than 0, then it was considered to be “marked”, otherwise the pixel was determined to be “empty”.

The first Naive Bayes model was created to have a LaPlace factor of 1, and to not use the kernel. The 3-fold cross validation accuracy was 0.82 and the accuracy on the test set was 0.81.

#Getting only the labels  
label <- train\_df$label  
#Creating a new daa frame with only those columns  
train\_nb <- train\_df[,2:ncol(train\_df)]  
  
#Discretizing the columns  
for (col in colnames(train\_nb)){  
 train\_nb[,col] <- discretize(train\_nb[,col], method = "fixed", breaks = c(-1, 0.1, 256),  
 labels = c("empty", "mark"))  
}  
#Counting number of columns with a marking  
count\_cols <- colSums(train\_nb == "mark")  
#Only columns with 10000 or more marks  
best\_cols <- names(count\_cols[count\_cols > 10000])  
#Filtering onlly thise columns  
train\_nb <- train\_nb[,best\_cols]  
  
#Final training data frame  
train\_nb <- data.frame(label = label, train\_nb)  
#Fixing the test data frame  
test\_nb <- test\_df[,2:ncol(test\_df)]  
#Getting only the same columns in training set  
test\_nb <- test\_nb[,best\_cols]  
#Discretizing the test set  
for (col in colnames(test\_nb)){  
 test\_nb[,col] <- discretize(test\_nb[,col], method = "fixed", breaks = c(-1, 0.1, 256),  
 labels = c("empty", "mark"))  
}  
  
#getting test labels  
test\_label <- test\_df$label  
#creating the test data frame  
test\_nb <- data.frame(label = test\_label, test\_nb)  
#training the model  
nb\_model <- train(label ~ ., data = train\_nb, tuneGrid =   
 data.frame(fL = 1, adjust = 1, usekernel = F),   
 trControl = trainControl(method = "cv", number = 3),  
 method = "nb")

nb\_model

## Naive Bayes   
##   
## 33604 samples  
## 233 predictor  
## 10 classes: '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'   
##   
## No pre-processing  
## Resampling: Cross-Validated (3 fold)   
## Summary of sample sizes: 22404, 22402, 22402   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.8234435 0.8038041  
##   
## Tuning parameter 'fL' was held constant at a value of 1  
## Tuning  
## parameter 'usekernel' was held constant at a value of FALSE  
##   
## Tuning parameter 'adjust' was held constant at a value of 1

#Predicting the test set  
preds <- suppressWarnings(predict(nb\_model, test\_nb, type = "raw"))  
#Confusion matrix  
confusionMatrix(preds, test\_df$label)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1 2 3 4 5 6 7 8 9  
## 0 735 0 8 6 2 18 3 4 1 7  
## 1 0 864 17 9 0 5 4 13 18 3  
## 2 10 30 649 37 14 21 30 38 27 14  
## 3 1 5 14 638 0 129 2 0 37 17  
## 4 2 1 20 0 699 11 13 15 8 81  
## 5 64 27 14 69 16 524 45 16 49 16  
## 6 5 3 42 10 4 20 721 0 9 2  
## 7 1 0 8 7 4 4 0 726 2 33  
## 8 8 6 58 70 26 18 9 12 642 26  
## 9 0 0 5 24 49 9 0 56 19 638  
##   
## Overall Statistics  
##   
## Accuracy : 0.8142   
## 95% CI : (0.8057, 0.8225)  
## No Information Rate : 0.1115   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7935   
## Mcnemar's Test P-Value : NA

A Naive Bayes model was also created with a heavier regularization parameter with an fL of 50 was created to see if there was any overfitting in the problem. The new model had the same cross validation accuracy of 0.82; however the prediction on the test set had the same accuarcy as the Naive Bayes Laplace factor.

#training the model  
nb2 <- suppressWarnings(train(label ~ ., data = train\_nb, tuneGrid =   
 data.frame(fL = 50, adjust = 1, usekernel = F),   
 trControl = trainControl(method = "cv", number = 3),  
 method = "nb"))  
#Looking at the accuracy of the training model   
nb2

## Naive Bayes   
##   
## 33604 samples  
## 233 predictor  
## 10 classes: '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'   
##   
## No pre-processing  
## Resampling: Cross-Validated (3 fold)   
## Summary of sample sizes: 22402, 22403, 22403   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.8236765 0.8040474  
##   
## Tuning parameter 'fL' was held constant at a value of 50  
## Tuning  
## parameter 'usekernel' was held constant at a value of FALSE  
##   
## Tuning parameter 'adjust' was held constant at a value of 1

#Predicting the test set  
preds2 <- suppressWarnings(predict(nb2, test\_nb, "raw"))  
#Confusion matric for the test set  
confusionMatrix(preds2, test\_nb$label)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1 2 3 4 5 6 7 8 9  
## 0 735 0 8 6 2 18 3 4 1 7  
## 1 0 864 17 9 0 5 4 13 18 3  
## 2 10 30 649 37 14 21 30 38 27 14  
## 3 1 5 14 638 0 129 2 0 37 17  
## 4 2 1 20 0 699 11 13 15 8 81  
## 5 64 27 14 69 16 524 45 16 49 16  
## 6 5 3 42 10 4 20 721 0 9 2  
## 7 1 0 8 7 4 4 0 726 2 33  
## 8 8 6 58 70 26 18 9 12 642 26  
## 9 0 0 5 24 49 9 0 56 19 638  
##   
## Overall Statistics  
##   
## Accuracy : 0.8142   
## 95% CI : (0.8057, 0.8225)  
## No Information Rate : 0.1115   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7935   
## Mcnemar's Test P-Value : NA

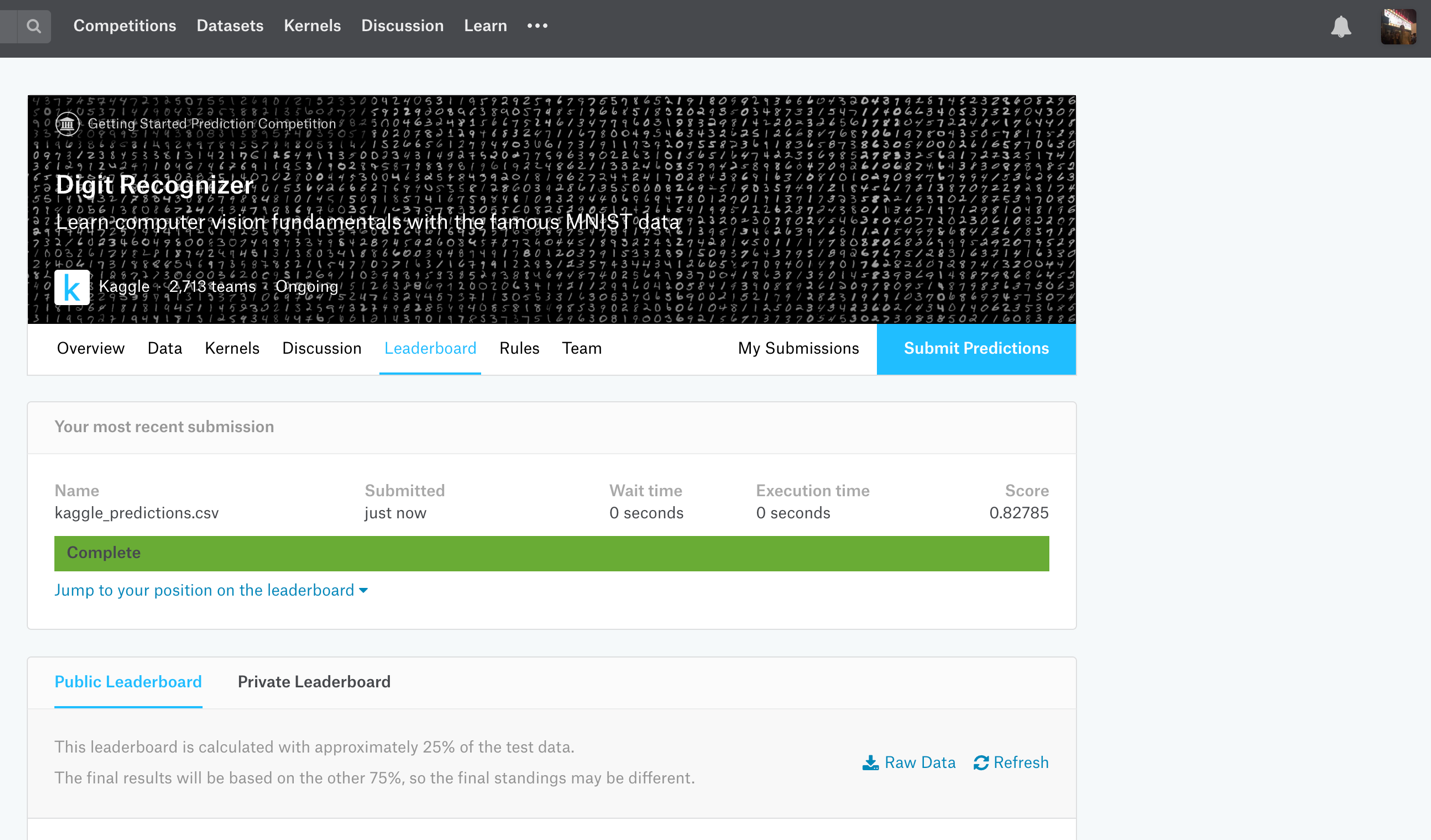
## Performance Comparison

When comparing the two models, the both performed similarly in terms of accuracy on the test set; however the tuned decision tree model had the best overall accuracy on the test set with an accuracy of 0.82. The decision tree model took a bit longer to train than the Naive Bayes model. It did take a bit longer to predict the naive bayes model compared to the decison tree model.

The decision tree model most likely took longer to train because there are so many different varaibles to parse through and split. The Naive Bayes model still took a good bit of time to train, but was definitely faster. This is likely because there are only a set number of calculations to work through, and didn’t have to double check each leaf to see if there were any better splits.

## Kaggle Results

Because the decision tree performed slightly better, I decided to use the decision tree model to predict the unknown data for the kaggle competition. A data frame was created to contain the image ID and the predicted class, and the row names were excluded so only image IDs and the classification was submitted. The final score was 0.82785, which was right within the range of my cross validation/test accuracies!



#Predictions with best model  
final\_preds <- predict(dt\_tuned, validation, "raw")  
#image ids  
ids <- seq\_along(final\_preds)  
#Saving in data frame  
predictions <- data.frame("ImageId" = ids, Label = final\_preds)  
#writing to file  
write.csv(predictions, "kaggle\_predictions.csv", row.names = F)