# **Building Digit Recognizer Models and Their Comparison** By Shivani S 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 Naïve Bayes classifier and decision tree algorithms and compare their accuracies.

## **Decision Tree**

The data processing does not involve many steps. 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.

3 fold cross validation is run on the model. The model runs for various complexity parameters. For every complexity parameter with 3 fold classification, the accuracy is obtained. Weighing the models based on the accuracy of their prediction, we choose the model with the highest accuracy. The model is then used to predict for the test dataset that the model has not encountered before. We check the model’s accuracy on the test dataset.

Tuning for Decision Tree We can tune the parameters of a decision tree

# set up 3-fold cross validation procedure  
train\_control <- trainControl(  
 method = "cv",   
 number = 3  
 )  
  
# more advanced option, run 3 fold cross validation 10 times  
train\_control\_adv <- trainControl(  
 method = "repeatedcv",   
 number = 3,  
 repeats = 10  
 )  
  
# set up tuning grid  
search\_grid <- expand.grid(.cp=c(0.01,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45))  
  
# train model  
dt.m1 <- train(  
 x = x,  
 y = y,  
 method = "rpart",  
 trControl = train\_control,  
 tuneGrid = search\_grid  
 )  
  
# top 5 modesl  
dt.m1$results %>%   
 top\_n(5, wt = Accuracy) %>%  
 arrange(desc(Accuracy))

## cp Accuracy Kappa AccuracySD KappaSD  
## 1 0.01 0.6167245 0.57383995 0.025847765 0.02874455  
## 2 0.05 0.4559367 0.39394018 0.011000983 0.01203960  
## 3 0.10 0.1410265 0.03434939 0.049958446 0.05949489  
## 4 0.15 0.1121964 0.00000000 0.000122433 0.00000000  
## 5 0.20 0.1121964 0.00000000 0.000122433 0.00000000  
## 6 0.25 0.1121964 0.00000000 0.000122433 0.00000000  
## 7 0.30 0.1121964 0.00000000 0.000122433 0.00000000  
## 8 0.35 0.1121964 0.00000000 0.000122433 0.00000000  
## 9 0.40 0.1121964 0.00000000 0.000122433 0.00000000  
## 10 0.45 0.1121964 0.00000000 0.000122433 0.00000000

# results for best model  
confusionMatrix(dt.m1)

## Cross-Validated (3 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction 0 1 2 3 4 5 6 7 8 9  
## 0 7.7 0.0 1.0 0.6 0.0 1.0 0.5 0.2 0.4 0.1  
## 1 0.0 9.4 1.2 0.4 0.3 0.4 0.6 0.1 0.9 0.1  
## 2 0.1 0.2 4.1 0.4 0.2 0.2 0.5 0.5 0.4 0.1  
## 3 0.3 0.3 0.3 5.7 0.1 1.0 0.3 0.0 0.8 0.1  
## 4 0.0 0.3 0.2 0.1 6.2 0.3 0.6 0.3 0.2 0.3  
## 5 0.7 0.1 0.4 1.2 0.5 4.4 0.7 0.2 1.2 1.4  
## 6 0.0 0.1 0.8 0.1 0.3 0.5 5.4 0.0 0.3 0.1  
## 7 0.7 0.3 0.4 0.3 1.0 0.5 0.5 7.9 0.3 0.9  
## 8 0.3 0.4 1.2 0.6 0.4 0.7 0.6 0.4 5.3 0.2  
## 9 0.1 0.0 0.2 0.6 1.0 0.3 0.1 0.6 0.7 5.5  
##   
## Accuracy (average) : 0.6167

pred <- predict(dt.m1, newdata = digit\_test)  
confusionMatrix(pred, digit\_test$label)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1 2 3 4 5 6 7 8 9  
## 0 331 3 37 18 1 36 25 2 7 2  
## 1 2 401 27 10 9 10 19 4 34 3  
## 2 0 23 225 9 0 1 1 25 8 0  
## 3 7 23 19 283 2 53 12 7 32 11  
## 4 0 8 13 8 226 13 33 10 10 8  
## 5 15 2 7 57 27 163 52 25 63 66  
## 6 3 0 20 1 13 41 189 8 23 13  
## 7 45 14 23 12 58 32 43 353 10 62  
## 8 9 4 37 26 11 36 24 11 170 14  
## 9 2 0 12 22 57 3 6 20 36 207  
##   
## Overall Statistics  
##   
## Accuracy : 0.607   
## 95% CI : (0.592, 0.6218)  
## No Information Rate : 0.1139   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5627   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.79952 0.83891 0.53571 0.63453 0.55941 0.42010  
## Specificity 0.96538 0.96828 0.98227 0.95576 0.97285 0.91759  
## Pos Pred Value 0.71645 0.77264 0.77055 0.63029 0.68693 0.34172  
## Neg Pred Value 0.97778 0.97907 0.95008 0.95652 0.95399 0.93953  
## Prevalence 0.09862 0.11386 0.10005 0.10624 0.09624 0.09242  
## Detection Rate 0.07885 0.09552 0.05360 0.06741 0.05384 0.03883  
## Detection Prevalence 0.11005 0.12363 0.06956 0.10696 0.07837 0.11363  
## Balanced Accuracy 0.88245 0.90360 0.75899 0.79514 0.76613 0.66884  
## Class: 6 Class: 7 Class: 8 Class: 9  
## Sensitivity 0.46782 0.75914 0.43257 0.53627  
## Specificity 0.96784 0.91990 0.95480 0.95855  
## Pos Pred Value 0.60772 0.54141 0.49708 0.56712  
## Neg Pred Value 0.94469 0.96842 0.94217 0.95330  
## Prevalence 0.09624 0.11077 0.09362 0.09195  
## Detection Rate 0.04502 0.08409 0.04050 0.04931  
## Detection Prevalence 0.07408 0.15531 0.08147 0.08695  
## Balanced Accuracy 0.71783 0.83952 0.69368 0.74741

As seen from the analysis, the model with the highest training accuracy is the model with complexity parameter of 0.01. The accuracy on the training set is 0.6167. On testing this model with the test data, the accuracy comes out to be 0.607, which is closer to the training accuracy.

## **Naïve Bayes Classifier**

For the data processing step, we do not need to do anything beyond the transformation of the target variable from integer type to factor type to make the model understand that classification with 10 levels from 0-9 is taking place.

3 fold cross validation is run on the model. The model runs for various hyperparameters. For every hyperparameter combination with 3 fold classification, the accuracy is obtained. The laplace and the adjust parameter is given values 0, 1 and 0, 1, 2 respectively. Weighing the models based on the accuracy of their prediction, we choose the model with the highest accuracy. The model is then used to predict for the test dataset that the model has not encountered before. We check the model’s accuracy on the test dataset.

Tuning for Naive Bayes Classifier

# set up 3-fold cross validation procedure  
train\_control <- trainControl(  
 method = "cv",   
 number = 3  
 )  
  
# more advanced option, run 3 fold cross validation 10 times  
train\_control\_adv <- trainControl(  
 method = "repeatedcv",   
 number = 3,  
 repeats = 10  
 )  
  
# set up tuning grid  
search\_grid <- expand.grid(usekernel = c(FALSE),  
 laplace = c(0, 1),   
 adjust = c(0,1,2))  
  
# train model  
nb.m1 <- train(  
 x = x,  
 y = y,  
 method = "naive\_bayes",  
 trControl = train\_control,  
 tuneGrid = search\_grid  
 )

# top 5 modesl  
nb.m1$results %>%   
 top\_n(5, wt = Accuracy) %>%  
 arrange(desc(Accuracy))

## usekernel laplace adjust Accuracy Kappa AccuracySD KappaSD  
## 1 FALSE 0 0 0.4502117 0.3896174 0.01012612 0.01113708  
## 2 FALSE 0 1 0.4502117 0.3896174 0.01012612 0.01113708  
## 3 FALSE 0 2 0.4502117 0.3896174 0.01012612 0.01113708  
## 4 FALSE 1 0 0.4502117 0.3896174 0.01012612 0.01113708  
## 5 FALSE 1 1 0.4502117 0.3896174 0.01012612 0.01113708  
## 6 FALSE 1 2 0.4502117 0.3896174 0.01012612 0.01113708

# results for best model  
confusionMatrix(nb.m1)

## Cross-Validated (3 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction 0 1 2 3 4 5 6 7 8 9  
## 0 9.0 0.0 2.0 3.1 0.6 2.5 0.4 0.5 0.5 0.1  
## 1 0.0 11.0 1.0 1.5 0.5 1.4 0.6 0.7 2.8 0.7  
## 2 0.0 0.0 1.5 0.0 0.0 0.0 0.0 0.0 0.1 0.0  
## 3 0.0 0.0 0.4 1.8 0.0 0.0 0.0 0.0 0.0 0.0  
## 4 0.0 0.0 0.0 0.0 0.3 0.0 0.0 0.0 0.0 0.0  
## 5 0.0 0.0 0.1 0.0 0.1 0.5 0.0 0.0 0.0 0.0  
## 6 0.2 0.1 2.7 0.5 0.6 0.3 7.9 0.0 0.1 0.1  
## 7 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.5 0.0 0.0  
## 8 0.2 0.1 1.6 1.5 0.8 2.8 0.1 0.5 3.8 0.1  
## 9 0.5 0.1 0.4 1.6 7.1 1.6 0.9 7.2 3.1 7.7  
##   
## Accuracy (average) : 0.4502

pred <- predict(nb.m1, newdata = digit\_test)  
confusionMatrix(pred, digit\_test$label)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1 2 3 4 5 6 7 8 9  
## 0 397 3 87 150 26 142 22 28 34 7  
## 1 3 464 27 67 11 33 10 25 106 18  
## 2 2 1 81 6 0 4 0 0 1 1  
## 3 0 0 18 72 2 3 0 2 1 0  
## 4 0 0 1 1 10 1 1 2 2 1  
## 5 1 0 3 0 7 25 1 2 7 1  
## 6 0 4 112 16 16 10 332 3 5 0  
## 7 0 0 1 1 0 0 0 71 0 2  
## 8 1 2 75 74 38 117 8 10 144 6  
## 9 10 4 15 59 294 53 30 322 93 350  
##   
## Overall Statistics  
##   
## Accuracy : 0.4636   
## 95% CI : (0.4484, 0.4788)  
## No Information Rate : 0.1139   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4047   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.95894 0.9707 0.19286 0.16143 0.024752 0.064433  
## Specificity 0.86813 0.9194 0.99603 0.99307 0.997628 0.994226  
## Pos Pred Value 0.44308 0.6073 0.84375 0.73469 0.526316 0.531915  
## Neg Pred Value 0.99485 0.9959 0.91736 0.90878 0.905719 0.912551  
## Prevalence 0.09862 0.1139 0.10005 0.10624 0.096236 0.092425  
## Detection Rate 0.09457 0.1105 0.01929 0.01715 0.002382 0.005955  
## Detection Prevalence 0.21343 0.1820 0.02287 0.02334 0.004526 0.011196  
## Balanced Accuracy 0.91353 0.9450 0.59444 0.57725 0.511190 0.529329  
## Class: 6 Class: 7 Class: 8 Class: 9  
## Sensitivity 0.82178 0.15269 0.36641 0.90674  
## Specificity 0.95625 0.99893 0.91301 0.76915  
## Pos Pred Value 0.66667 0.94667 0.30316 0.28455  
## Neg Pred Value 0.98054 0.90444 0.93312 0.98787  
## Prevalence 0.09624 0.11077 0.09362 0.09195  
## Detection Rate 0.07909 0.01691 0.03430 0.08337  
## Detection Prevalence 0.11863 0.01787 0.11315 0.29300  
## Balanced Accuracy 0.88901 0.57581 0.63971 0.83794

As seen from the analysis, the different values of Laplace and adjust give the same accuracy of 0.45. The accuracy on the testing set for Laplace 0 and adjust 0 is 0.463. The accuracy for this case of testing seems to have increased, although that could be a coincidence.

## **Results**

From the two models, decision trees gave a better accuracy of 0.607 as compared to 0.45 of Naïve Bayes classifier. In data science, there is no single shoe that fits all, however, in the case of digit recognizer with the data we have, decision trees prove to be better fitting. The decision trees algorithm is much faster for prediction at 1.3 seconds of processing time as compared to Naïve Bayes classifier which takes 4.3 seconds processing time.

Decision Trees can be less of a black box than Naïve Bayes and hence can prove to be easier to tune. In Naïve Bayes we could not see any distinction even after changing the parameters.