**Introduction**

The rise of artificial intelligence technology along with machine and deep learning are opening limitless possibilities. Handwriting recognition technology through artificial neural networks is helping the build platforms that are reflective of the human experience. Handwriting recognition is the ability of a computer to receive and interpret intelligible handwritten input from sources such as paper documents, photographs, touch-screens and other devices. The ability to interpret the meaning of users handwritten input in real-time is teaching machines to adapt to the user.

Handwriting recognizers are divided into two categories—on-line and off-line. On-line recognizers run, and receive the data, as the user writes. They must process and recognize the handwriting in real- to near real-time. The surface used for handwriting is usually a tablet and it is used along with a digital pen to write on the surface. As the pen moves across the surface, the two-dimensional co-ordinates of successive points are collected and stored as a function of time. Off-line recognizers run after the data have been collected, and the image of the handwriting, for analysis, is given to the recognizer as a bitmap. Thus, the speed of the recognizer is not dependent on the writing speed of the user, but the speed dictated by the speciﬁcations of the system, in words or characters per second. The off-line recognizers are suitable for automatic conversion of paper documents to electric documents which then may be interpreted by computers.

**About the Data**

The data set comes from the Kaggle Digit Recognizer competition. The goal is to recognize digits 0 to 9 in handwriting images. Because the original data set is too large to be loaded in Weka GUI, the data for the assignment was systematically sampled, 10% of the data by selecting the 10th, 20th examples and so on.

The competition data files train.csv and test.csv contain gray-scale images of hand-drawn digits, from zero through nine. Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255, inclusive.

The training data set, (train.csv), has 785 columns. The first column, called "label", is the digit that was drawn by the user. The rest of the columns contain the pixel-values of the associated image.

Each pixel column in the training set has a name like pixelx, where x is an integer between 0 and 783, inclusive. To locate this pixel on the image, suppose that we have decomposed x as x = i \* 28 + j, where i and j are integers between 0 and 27, inclusive. Then pixelx is located on row i and column j of a 28 x 28 matrix, (indexing by zero).

**Analysis**

The data has 1400 rows and 785 columns. There are no missing values or duplicate records. The distribution is consistent:

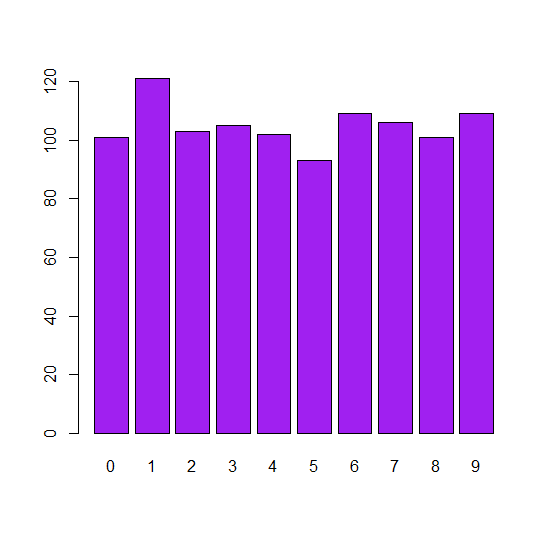
0 1 2 3 4 5 6 7 8 9

142 167 130 140 131 127 145 137 137 144

75% of the data is put into a training set and consists of 1050 rows.

0 1 2 3 4 5 6 7 8 9

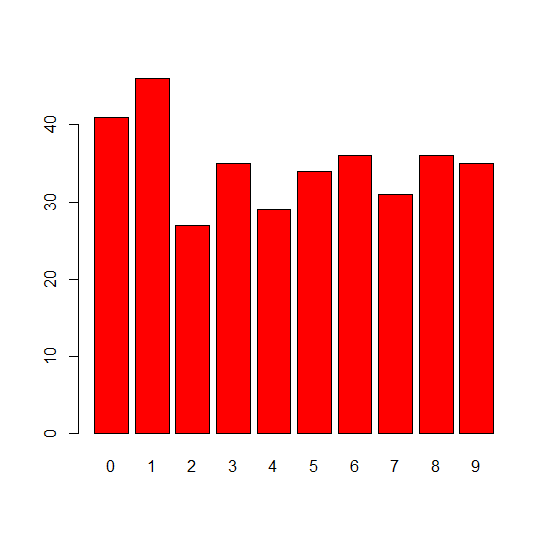
101 121 103 105 102 93 109 106 101 109



The remaining 350 rows are put into a test set.

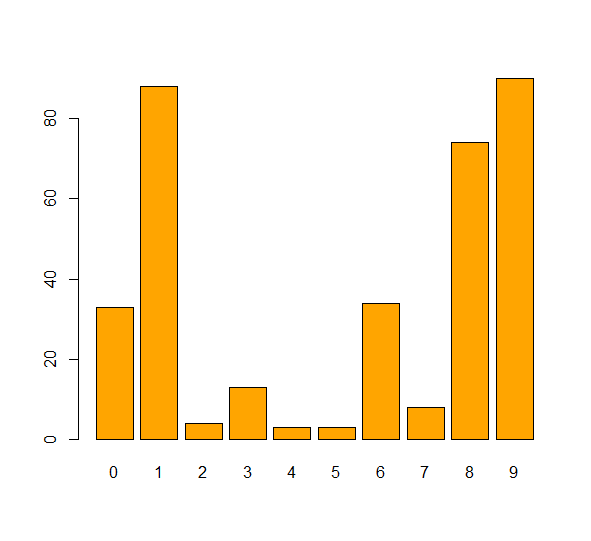
0 1 2 3 4 5 6 7 8 9

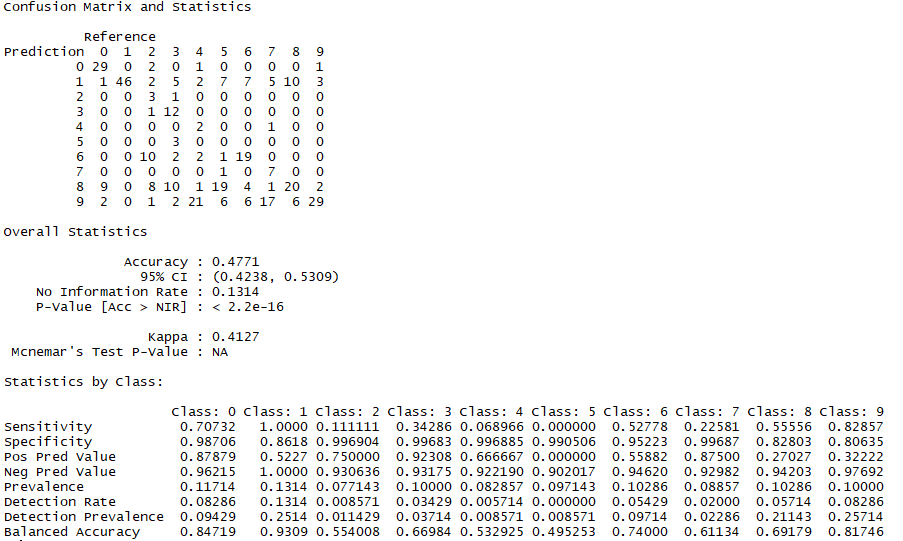
41 46 27 35 29 34 36 31 36 35



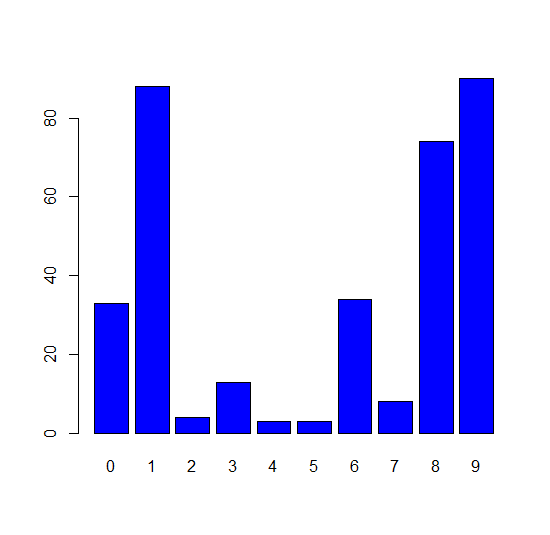
The label column is removed and placed into a new set for comparing predicted vs. actuals.

NB\_object model with default settings





NB\_e1071 model laplace = 3



Confusion Matrix and Statistics

Reference

Prediction 0 1 2 3 4 5 6 7 8 9

0 29 0 2 0 1 0 0 0 0 1

1 1 46 2 5 2 7 7 5 10 3

2 0 0 3 1 0 0 0 0 0 0

3 0 0 1 12 0 0 0 0 0 0

4 0 0 0 0 2 0 0 1 0 0

5 0 0 0 3 0 0 0 0 0 0

6 0 0 10 2 2 1 19 0 0 0

7 0 0 0 0 0 1 0 7 0 0

8 9 0 8 10 1 19 4 1 20 2

9 2 0 1 2 21 6 6 17 6 29

Overall Statistics

Accuracy : 0.4771

95% CI : (0.4238, 0.5309)

No Information Rate : 0.1314

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.4127

Mcnemar's Test P-Value : NA

Statistics by Class:

Class: 0 Class: 1 Class: 2 Class: 3 Class: 4

Sensitivity 0.70732 1.0000 0.111111 0.34286 0.068966

Specificity 0.98706 0.8618 0.996904 0.99683 0.996885

Pos Pred Value 0.87879 0.5227 0.750000 0.92308 0.666667

Neg Pred Value 0.96215 1.0000 0.930636 0.93175 0.922190

Prevalence 0.11714 0.1314 0.077143 0.10000 0.082857

Detection Rate 0.08286 0.1314 0.008571 0.03429 0.005714

Detection Prevalence 0.09429 0.2514 0.011429 0.03714 0.008571

Balanced Accuracy 0.84719 0.9309 0.554008 0.66984 0.532925

Class: 5 Class: 6 Class: 7 Class: 8 Class: 9

Sensitivity 0.000000 0.52778 0.22581 0.55556 0.82857

Specificity 0.990506 0.95223 0.99687 0.82803 0.80635

Pos Pred Value 0.000000 0.55882 0.87500 0.27027 0.32222

Neg Pred Value 0.902017 0.94620 0.92982 0.94203 0.97692

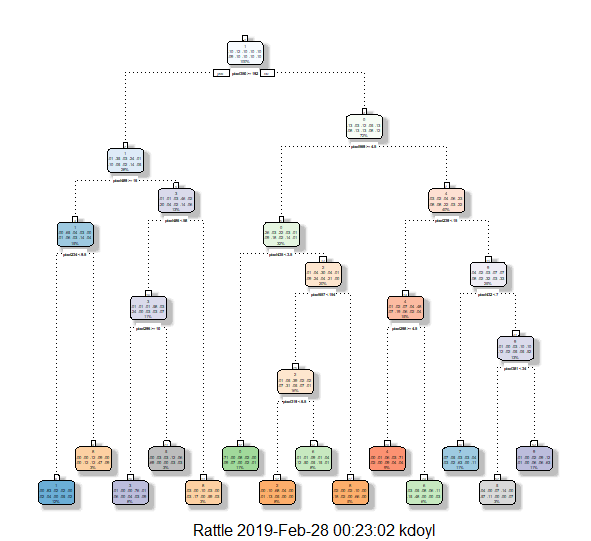
Prevalence 0.097143 0.10286 0.08857 0.10286 0.10000

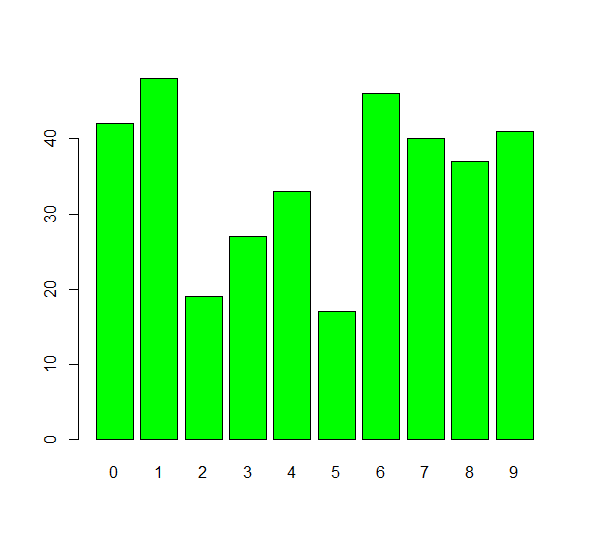
Detection Rate 0.000000 0.05429 0.02000 0.05714 0.08286

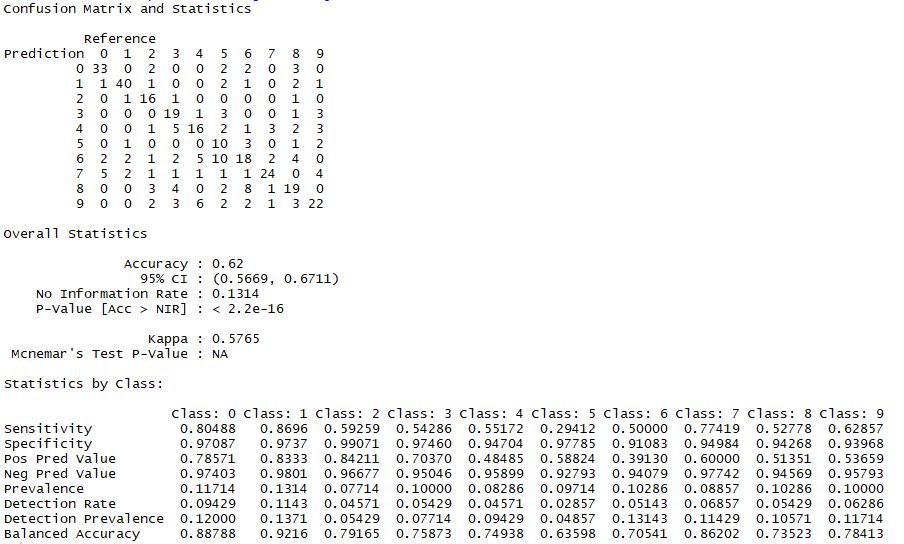
Detection Prevalence 0.008571 0.09714 0.02286 0.21143 0.25714

Balanced Accuracy 0.495253 0.74000 0.61134 0.69179 0.81746

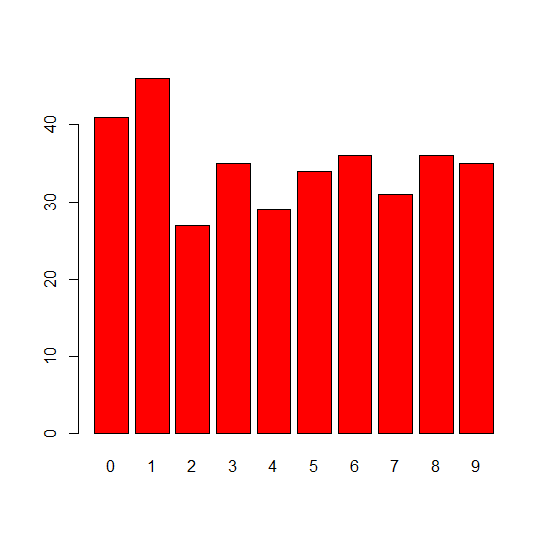
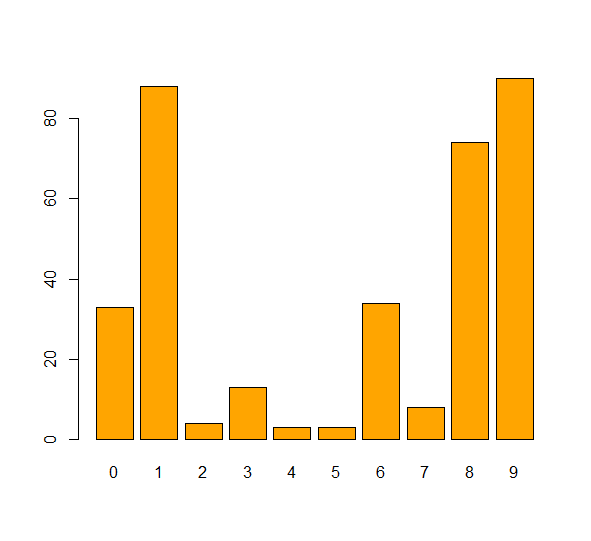
Decision Tree Model



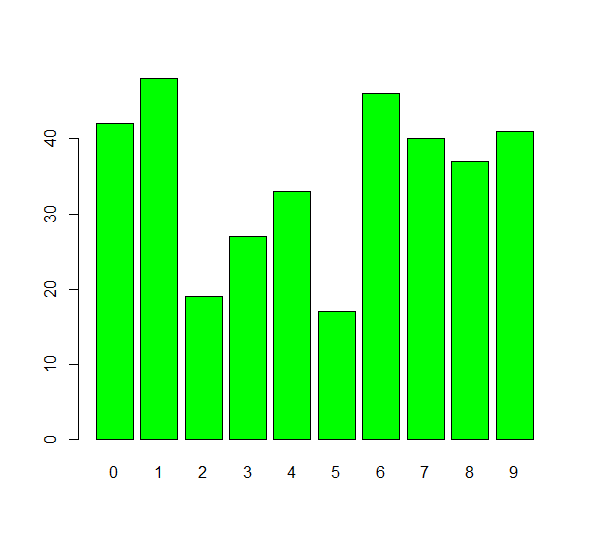




**Results**

Actual NaiveBayes Models

Decision Tree



The decision tree produced predicted results with 62% accuracy. The decision tree produced results slightly more quickly. The Naïve Bayes models produced predicted results with 48% accuracy. The Naïve Bayes models produced the same results with e1071 and naïve\_bayes algorithms. They produced the same results even when editing laplace values. The model did not correctly identify numbers 2 through 7. It appears they were misclassified as 1, 8, or 9.

**Conclusion**

The poor results were likely due to the small sample of the original data selected. The original Kaggle file was too large and would not load on the PC. The errors were intuitive. It makes sense for number to be misclassified as 1, 8, or 9.

The results made predictions better than a random guess, but not accurate enough for handwriting recognizers. Most handwriting recognizers are 87-93% accurate. With current hand-writing recognition technologies, recognition systems need to provide accurate recognition. The analyses of errors using confusion matrices show problems. These matrices also reveal opportunities for designers of handwriting products to adjust the algorithms to correct specific deficiencies in their recognizers. Handwriting recognition technology can benefit from adaptive algorithms; however, improving the learner experience with the technology may be the most important factor in overall user acceptance.