Introduction:

The ability to problem solve at an extremely high level is what separates humans extensively from other creatures on this earth. The human mind has adapted to be able to simulate different scenarios prior starting a problem to evaluate the best approach. While the magnitude of the human mind is awe-inspiring, it has its limitations. Comically, the approach to solving this problem was to create a machine as an extension of the human mind but with significantly more processing power and speed, enter the computer. As with most inventions, there are aspects that are left wanting. Computers process at a high level but, they abstract things poorly as they interpret literally. Thus, the prime partnership is a computer’s processing power paired with a human’s creativity. From this dynamic duo machine learning algorithms were born, and the term “artificial intelligence” was coined.

The goal of this exercise is to partake on a lesser scale, in the Kaggle Digit Recognizer competition. The classification problem at hand is attempting to successfully recognize hand-drawn digits by a user from 0-9. The original data set was too cumbersome, as such, a prorated version will be used for this exercise. The sampled data provided was bifurcated into training and testing sets and will be used within the Naïve Bayes and Decision Tree algorithms. The parameters will be tuned, and efficacy will be measured by a form of cross validation. A final decision will be made about which algorithm is superior in this instance.

Analysis and Models:

About the Data:

The data set is comprised of two csv files. The first file is “Kaggle-digit-train-sample-small-1400.csv” and the second is “Kaggle-digit-test.csv”. Both files contain hand drawn gray-scales images of numeric values beginning with 0 and ending at 9. Within in both files are images. Each image holds the dimensions 28x28 pixels yielding a pixel total of 784 for each image. Every pixel has a numeric association which denotes how light or dark that specific pixel is. The increasing numerical value indicates a concomitate increase in darkness. As such, lower value numbers will be lighter and higher numbered values will be darker. The pixel value range is from 0 to 255 inclusively.

The training data set (Kaggle-digit-train-sample-small-1400.csv) and the testing data set (Kaggle-digit-test.csv) are nearly identical. The only difference is the initial column in Kaggle-digit-train-sample-small-1400.csv, “label.” The training data has 785 variables. Column one is the “label” column which is the digit drawn by the user. Since the testing data excludes this column, it has one less row at 784. The rest of the columns in both data sets are the pixel values of the images. The pixel column names increment linearly from pixel1, pixel2, pixel3,…,pixel783. Being that the images are a 28x28 matrix the position and darkness of the pixel is correlated with the numerical image it represents. There are 28,000 observations in the testing data and 1400 in the training.

Regarding the extent of data preprocessing, the “label” column is converted to a factor or nominal variable in the testing data. The label column is also removed from the testing data. The total size of the dataset was reduced prior to starting. There are no missing values to remove. Below is a visualization of the training data with the dimensions on the x and y axis.

Graphical user interface, chart

Description automatically generated

Model:

The first algorithm used in R is NaïveBayes. The algorithm is based on Bayes theorem which is used to calculate the prior and conditional probabilities of outcomes. This package is an extension of the Naïve Bayes classifier in R. The function detects the class of each feature in the data set and assumes potential different distributions for each feature. Predictors are assumed to be independent within each class label or between every pair of features. This classifier tends to be well suited for cases of document classification or the filtering of spam. A small amount of training data is comparatively needed to estimate parameters. This result is a faster executing classifier than other options.

The second R model of choice for this analysis is the rpart model. This model is used for classification and regression decision trees. The data is separated into, a testing and a training set. The selected root node and internal nodes comprise the splitting attributes. A root node splits the data in half. If the root nodes lead to unanimous decision, they are regarded as pure. If the internal node subsets give inconsistent answers, they will be continually split until they attain a full decision. Multiple trees can fit the same data. There are two methods of splitting data. A two-way split creating a narrow and deep decision tree or, a multi-way split that creates a broad but shallow decision tree. The methodology is dictated by the problem and the type of variables contained in the data set. Pruning is essential for simplifying the tree by keeping only the most important splits. Accuracy is intentionally reduced to keep model usability ubiquitous and prevent overfitting.

The third R model used in this analysis is a Support Vector Machine. The SVM is a binary classifier that separates positive and negative examples. A unique aspect of this model is that it can solve both linearly separable and inseparable problems amongst different dimensions. The algorithm finds a linear hyperplane or decision boundary that separates the data. It seeks to maximize the margin between data points and fits the model directly in the middle. The number of support vectors indicates how complicated the model is. Prediction confidence comes in values. The farther the test data from the decision boundary is the predictor of confidence. The farther the better.

Results:

To maintain equitable results the default tuning parameters are pre-set. Additionally, all the same data and parameters are used for each algorithm. The three that will be compared are Naïve Bayes, Decision trees, and Support Vector Machines. A train control measure is implemented by way of a 3-fold cross validation process. This method is ideal for accounting for potential outliers. After the models are created a summary of model accuracy is generated.

Chart, scatter chart

Description automatically generated

The model shows Naïve Bayes and the Decision Tree to be the least accurate and SVM, Random Forest, and k-NN to be the most accurate. The accuracy of Naïve Bayes is ~20%, the accuracy of Decision Tree is ~50%, SVM, Random Forest, and k-NN are all in the ~90% accuracy range with SVM and Random Forest performing the best. The Kappa values, another measure of model performance almost exactly mirrors the accuracy measures.

The below chart for the classification and regression tree shows a breakdown of the complexity parameters (cp) and their respective accuracy and kappa values.

Graphical user interface, text, application

Description automatically generated

The chart is sorted in descending order by accuracy. There is no surprise that there is an inverse relationship between the complexity parameter and accuracy. The chart below visualized this relationship. It is quite evident that as the algorithm becomes more complex, it becomes less accurate. The maximal accuracy this model achieves in ~48%.

Chart, line chart

Description automatically generated

The next model is the Naïve Bayes. This algorithm does not generate a chart as its peers does. However, the initial chart shows the accuracy and kappa score which are both below 20%. Naïve Bayes was by far the lowest performing model of the 5. This could be attributed the the data set being too large for Naïve Bayes to handle properly.

The final and most accurate model used in this comparison is the SVM. Similarly, to the decision tree the below chart shows the accuracy and kappa score. However, it differs in that it has a column “C’ which represents the cost for misclassification the model is willing to pay. When C is large, the cost is high, and the model tries to build the model with the fewest training errors. Conversely, when C is small the margin is wider and more robust so less accurate. Unlike the Decision Tree and cost parameter, the high the C number is for the SVM the higher the accuracy will be as shown below.

Graphical user interface, text, application

Description automatically generated

The highest accuracy for the model is shown to be 92% with a C of 4 and a commensurate kappa of 92%. This relationship is further illustrated below.

Chart, line chart

Description automatically generated

Of the 3 models, the Support Vector Machine was by far the most accurate. Overall it was 44% more accurate than the Decision tree and 72% more accurate than Naïve Bayes model.

Conclusion:

The results of the amended Kaggle Digit Recognizer competition were that the Support Vector Machine reigned supreme over the Decision Tree classifier and Naïve Bayes model. The margin of dominance was not insignificant. Altering different parameters could potentially shift the balance of power amongst the algorithms. Additionally, the Naïve Bayes classifier could have performed better if the data was even further prorated. This would have also potentially reduced the efficacy of its competitors. Overall, looking at a snapshot of the data, SVM correctly predicted 5 out of the first 6 digits and incorrectly perceived the 0 to be a 4.