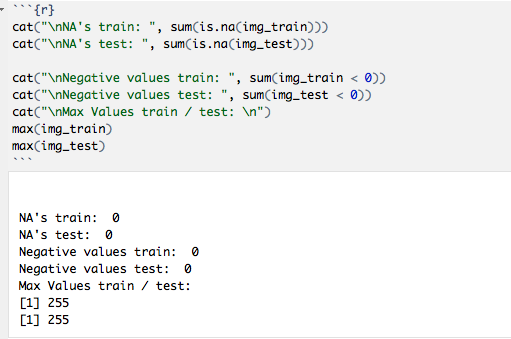
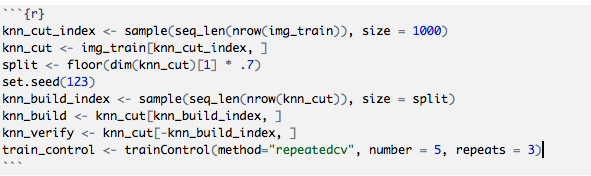
Homework 7 – Anthony Olivieri

# Introduction | Data Preparation

Typical data preparation techniques were used, but unnecessary. For example, evaluating the dataset for missing or outlier values returned no results. The dataset is complete and the values were all within a reasonable range.

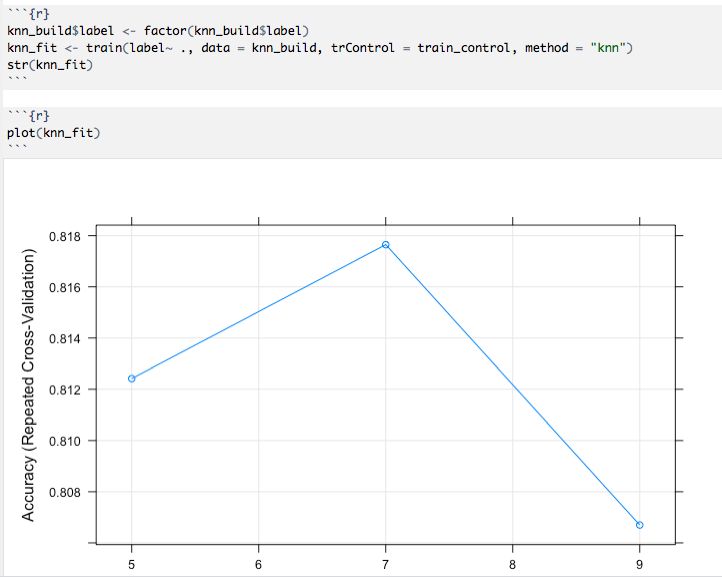


The kNN model, using full training data, was too processor intensive. The first attempts at kNN resulted in a crashed R Studio. The data was then stripped down to a thousand entries and that data was split into training and test data. While the new dataframes were called “knn\_build” and “knn\_verify” they were used for all models, not just kNN.



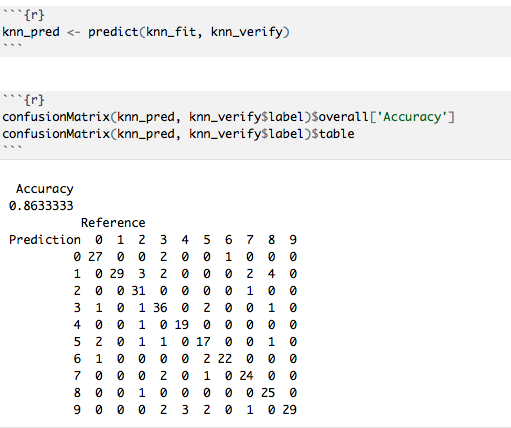
# Build and Cross-Validate kNN Model

The bottom line of code from the figure above shows a “train\_control” object being created for cross-validation. The number of folds created was five and the number of times run through was three. Anything larger was too processor intensive, but the added accuracy made up for the smaller training dataset of 700 rows.



The smaller cross-validated training set (called “knn\_build” in the above code) was used to train a kNN model through the caret function “train,” using the training\_control cross-validation. The fold’s relative accuracy per N neighbors is plotted above.

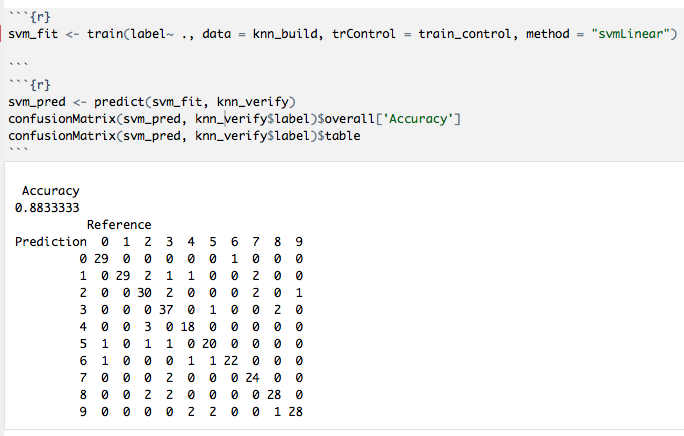
The accuracy was evaluated on a test set (called “knn\_verify”) by confusion matrix. The result was 86% accurate.



# SVM

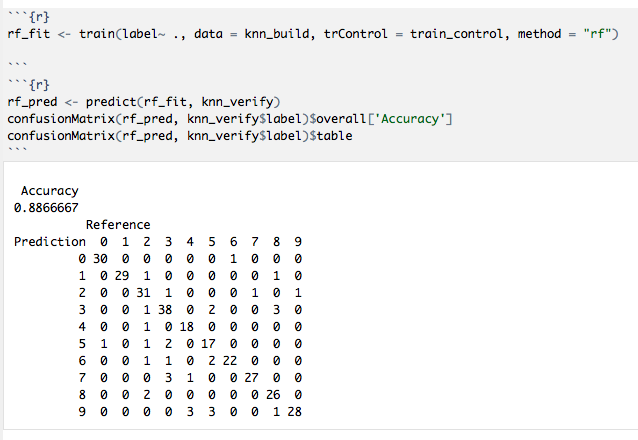
A Support Vector Machine model was created with the same smaller datasets and the same train\_control object. Preprocessing strategies of centering and scaling were used but lowered the overall accuracy and resulted in errors because so many of the pixel variables were all zeroes. Different kernels were used but resulted in lower accuracy. A linear SVM with no preprocessing, no kernel, and no tuneLength parameter yielded the highest accuracy when tested against the “verify” training data.

The accuracy tested at ~88%.



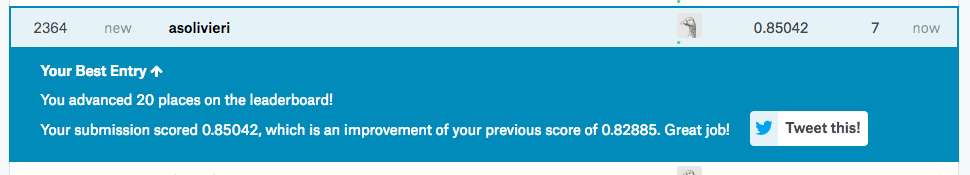
# Random Forest

Using the same smaller datasets and cross-validation train\_control object a Random Forest was model was created. Tuning of the Random Forest model by adjusting stepFactor and nTrees but resulted in lower accuracy. The random forest model was processor-intensive and required two restarts of R Studio so further experimentation was out-of-reach for this assignment. A straight random forest with no adjustment of parameters yielded a ~88% accuracy against training data.

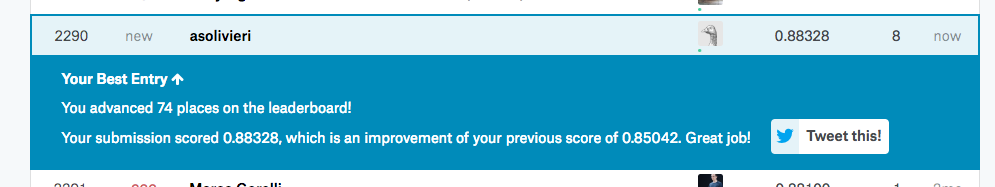


# Algorithm Performance

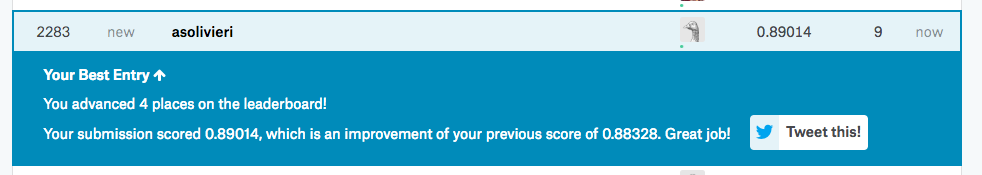
When checked against Kaggle for accuracy the kNN model achieved ~85%.



When checked against Kaggle for accuracy the SVM model achieved ~88%.



When checked against Kaggle for accuracy the Random Forest model achieved ~89%.



Two outcomes are noticeable for the SVM, kNN, and random forest models when compared against the previous models of naïve Bayes and a pruned decision tree. First, all of the models created this week (SVM, kNN, and random forest) outperformed the previous models. Second, the accuracy when tested against training data versus the accuracy when tested against Kaggle largely stayed the same.

This is a surprising result because the training data was cut down to just 1000 rows to save computing time. It is not a surprising result given that this week’s models were trained using cross-validation and last week’s models were not. The cross-validation of five folds at three repetitions yielded better accuracy against test data than a much larger dataset without cross-validation. The smaller dataset also cut down on processing power and computer time.

The random forest model performed best although it was the most processor intensive. One reason for the random forest model outperforming the others is that it is an ensemble learning method. Random forests are built from several decision trees and therefore accuracy increases. Coupled with the cross-validation the random forest model iterates through several possible classification outcomes and the best one is chosen.