**The Problem**

The model to be developed should be able to recognize hand drawn digits (0-9) from 28x28 Pixel images. The train data was a flat file containing 42000 rows (representing the images) and 785 columns (28x28+1 for the label) with 8bit gray values (0-255). The test data consisted of 28000 images in a similar format, the first column with the label was missing. The predictions of the labels of this test data is actually graded by Kaggle.

**The Approach**

I tried to reduce the complexity of the problem by reducing the number of variables. For all images in the test and training data column sums and row sums were determined. This made a total of 57 (28 rows +28 columns+1 label) columns in the new datasets. If successful, this approach should scale very good, since for an e.g. 100x100 pixel image the dimension size would be reduced from 10001 to 201.

Boxplots were the variance in the data for the individual digits are published here on git.

<https://github.com/JosefKnecht42/Kaggle/tree/master/digit-analyzer>

**The Methods**

The preparation step of calculating column and row means were done in Python due to much better performance and better debug possibilities. The resulting new train and test data was evaluated with R. Two models were considered, RandomForests and K-nearest Neighbour. Since RandomForests evaluate variables in isolation and pattern of variables appeared more appropriate the k-nearest Neighbour method using the FNN package was investigated further.

With the createDataPartition function from the Caret package various values for k and different distance functions were evaluated. However the accuracy should little difference from these parameter.

**The Results**

The benchmark supplied by kaggle gave an accuracy 0.96557. In this approach also the k nearest neighbour Method is applied on the unreduced dataset with 756 rows.

The score of the knn model based only on columns sums and row sums gave a score of 0.88186

The reduction of variables on the dataset comes apparently with the price of a worse model, even the computing time for one run was reduced by a factor of ~30.

**The Improvements**

I tried to improve the performance by

* Normalizing the column and row sums to 1 respectively
* By shifting the column and row sums so that the center of density if on row/column 14

These steps were performed again using Python.

Here again the knn model should little sensitivity to variation in the value of k or the distance function. The submission scored 0.88657 a very marginal improvement. Possibly the digit images were already positioned in the middle since for the case of “raw” images position adjustment should made some effect.