MNIST Project Report

Eric Mestiza

5/12/2019

Note: Report begins below the R code output.

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

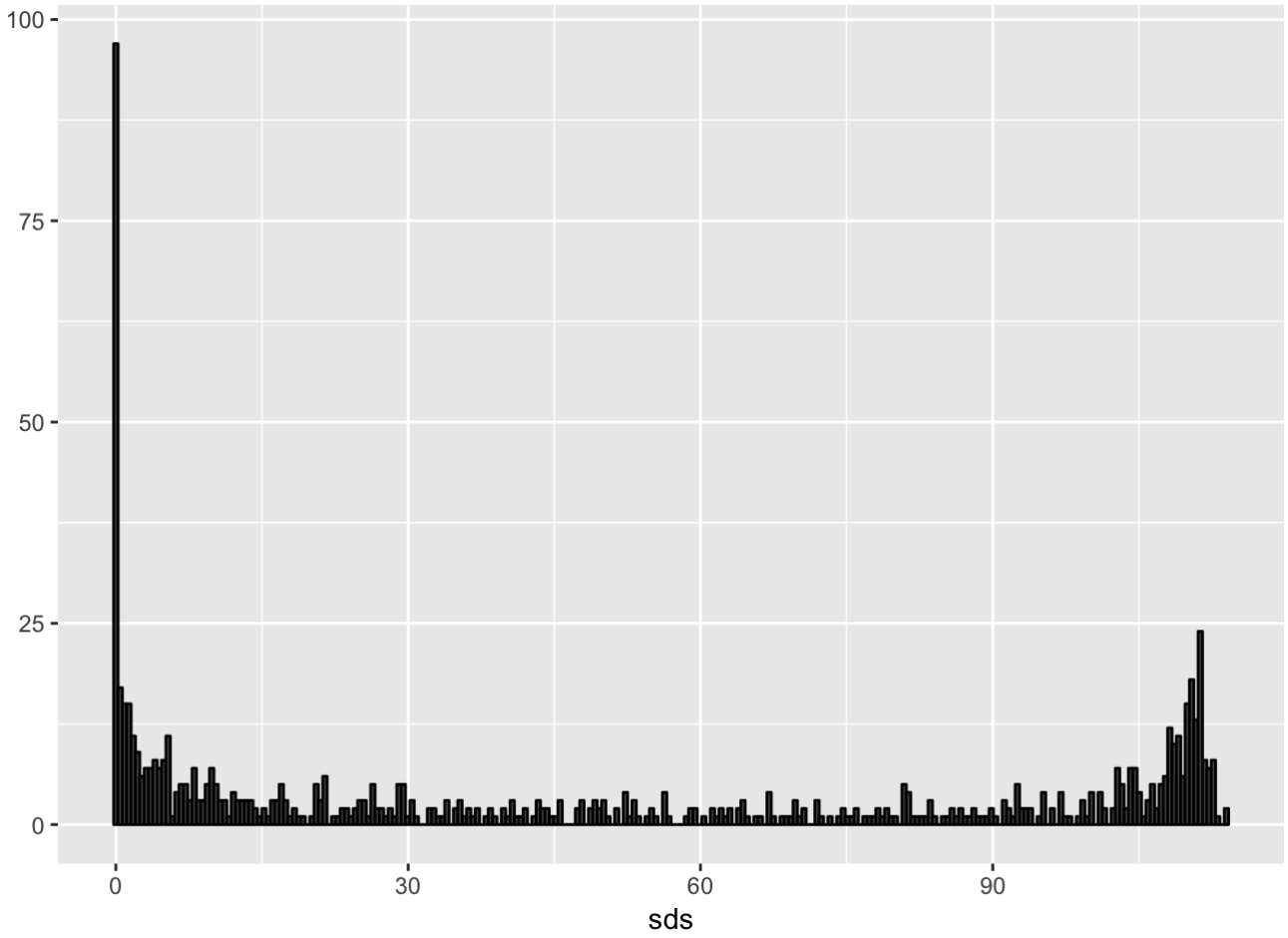
```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
## [1] "train" "test"
```

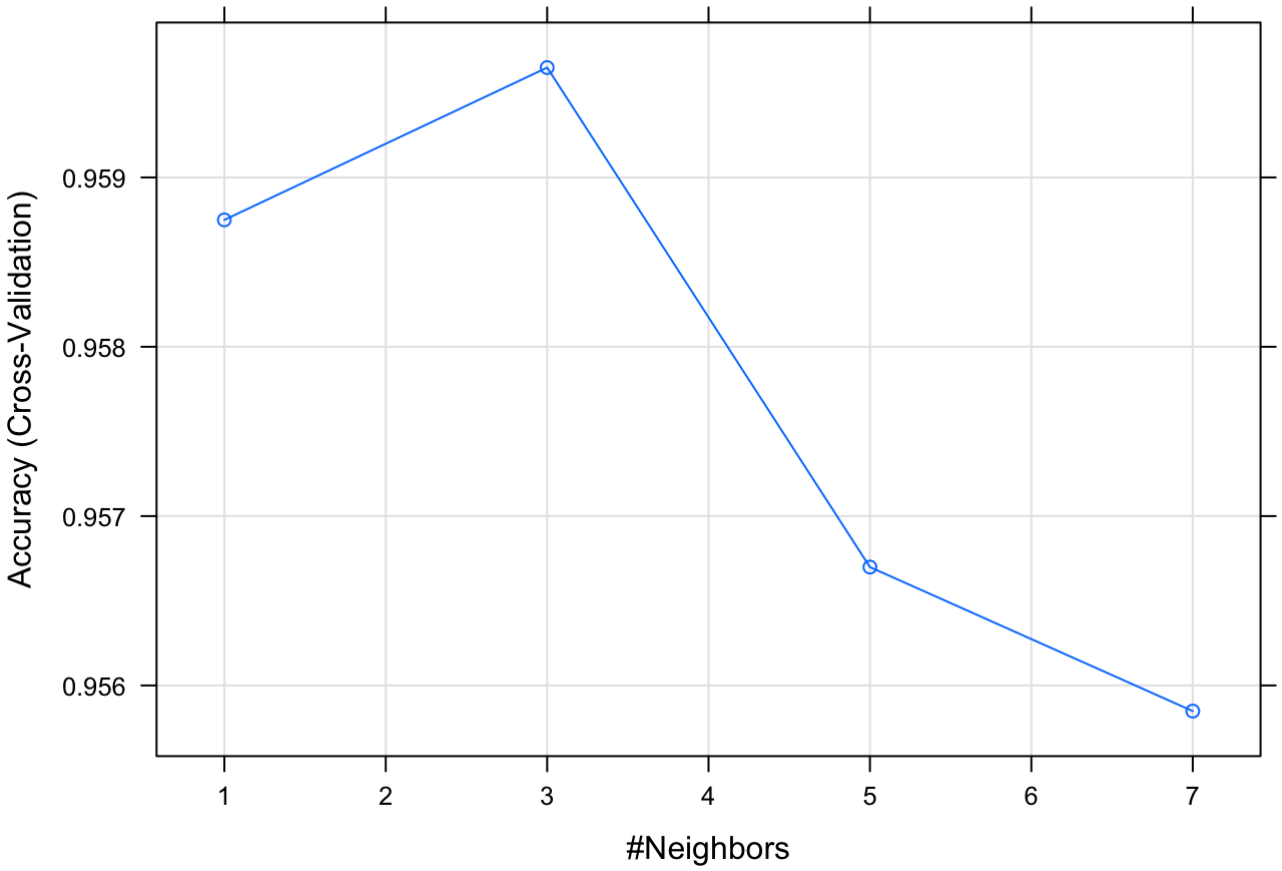
```
## [1] 60000   784
```

```
## [1] "integer"
```

```
##
##    0    1    2    3    4    5    6    7    8    9
## 5923 6742 5958 6131 5842 5421 5918 6265 5851 5949
```




```
## [1] 250
```




```
## Accuracy
##     0.97
```

```
##          Sensitivity Specificity
## Class: 0   1.0000000   0.9972558
## Class: 1   0.9905660   0.9960850
## Class: 2   0.9558824   0.9977728
## Class: 3   0.9500000   0.9972222
## Class: 4   0.9698492   0.9983343
## Class: 5   0.9767442   0.9961707
## Class: 6   0.9855072   0.9988846
## Class: 7   0.9955556   0.9915493
## Class: 8   0.9278351   0.9988926
## Class: 9   0.9473684   0.9944165
```

```
## Accuracy
##   0.9565
```

```
##          Sensitivity Specificity
## Class: 0   0.9814815   0.9977578
## Class: 1   1.0000000   0.9915398
## Class: 2   0.9554455   0.9949944
## Class: 3   0.9644670   0.9927898
## Class: 4   0.9425287   0.9978094
## Class: 5   0.9301075   0.9955899
## Class: 6   0.9848485   0.9966704
## Class: 7   0.9729730   0.9906336
## Class: 8   0.8871795   0.9994460
## Class: 9   0.9363636   0.9943820
```

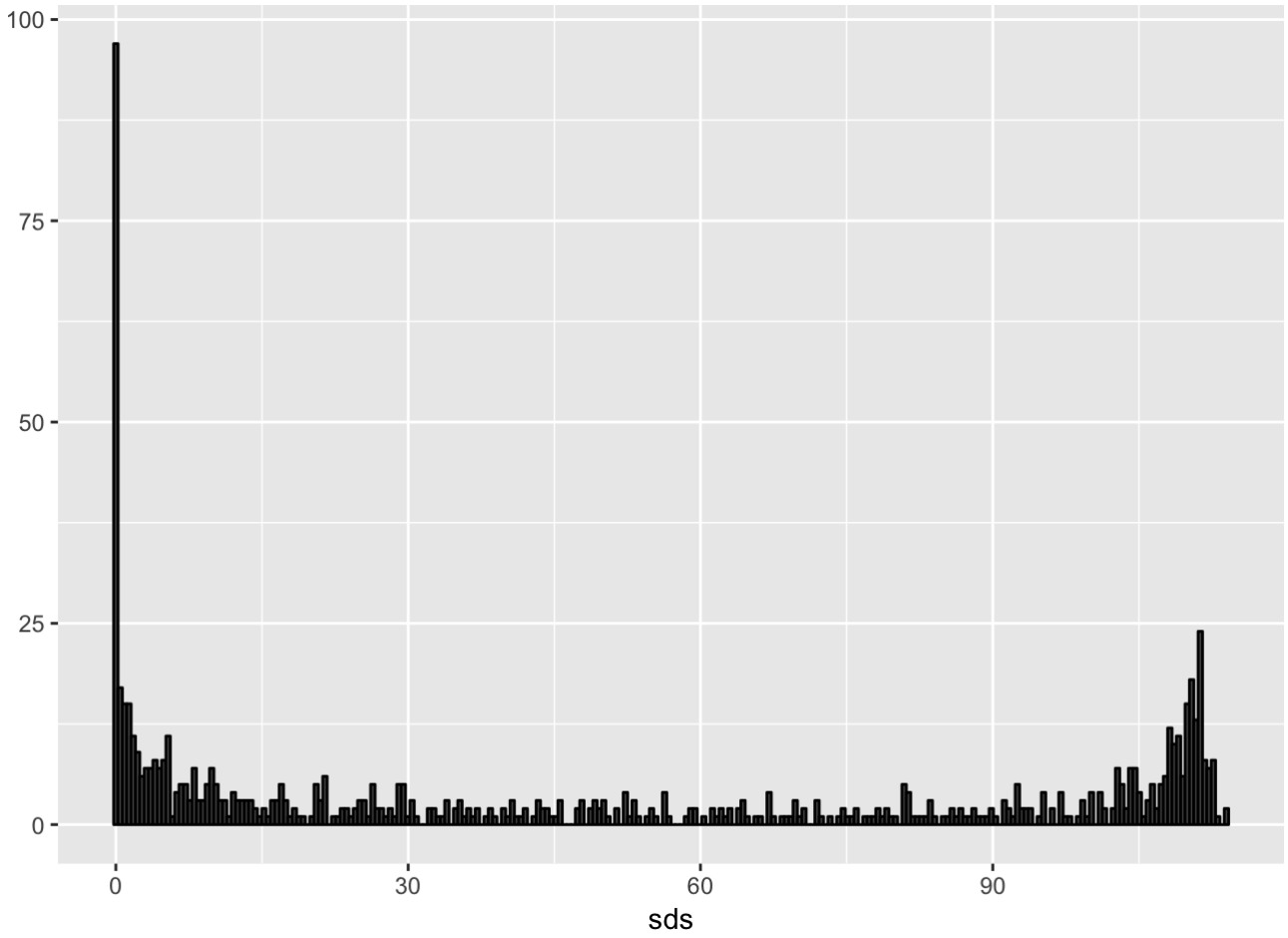
I. Introduction

The MNIST (Modified National Institute of Standards and Technology) dataset is a large dataset of handwritten digits that is used for training various image processing systems. The MNIST dataset is also a popular dataset used in machine learning competitions and will be used in this project. The goal of this project is to get highest accuracy possible in processing the images of digits from the MNIST dataset. To achieve this goal an algorithm will be developed using k-nearest neighbors (knn) algorithm to process the images. The k-nearest neighbors function is used for pattern recognition and is a non-parametric method used for classification or regression. A training dataset will be used to develop the algorithm and then the algorithm will be applied to the testing dataset. To determine the best model the accuracy parameter will be measured.

II. Methods and Analysis

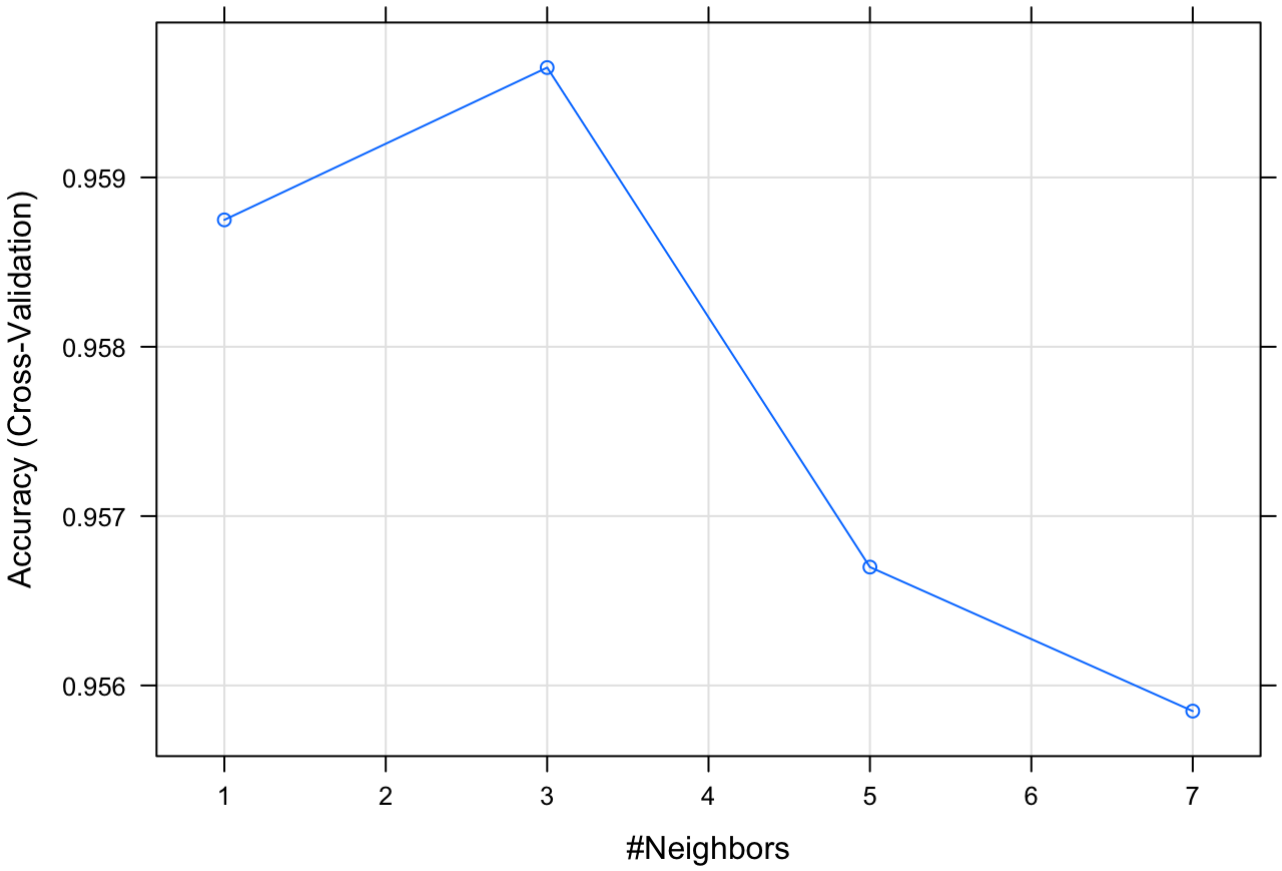
The MNIST dataset is loaded using the dslabs package. The dataset includes two components, a training set and a test set. Each of these components includes a matrix with features in the columns. To access these features use the dim( ) function. It also includes a vector with the classes as integers. To see this use the class( ) function. A smaller subset of the training dataset is used so it can run on a small computer in a short amount of time. In this project 20,000 random rows from the training set and 2,000 random rows from the test set are sampled.

An important step in machine learning is transforming predictors before running the machine learning algorithm. Another important step is removing predictors that are not useful. These steps are part of pre-processing. Examples of pre-processing include standardizing the predictors, transformation of predictors, removing predictors that are highly correlated with others, and removing predictors with very few non-unique values or close to zero variation. This project looks at the variability of the features. There are a large number of features with zero variability or almost zero variability. This code, sds <- colSds(x), can be used to compute the standard deviation of each column and then plot them in a histogram. The histogram is shown below.



This is expected, because there are parts of the image that rarely contain writing, very few dark pixels, so there's little variation and almost all the values are 0. The caret package includes a function that recommends features to be removed because of near zero variance. The following lines of code can be used: nzv <- nearZeroVar(x) and col_index <- setdiff(1:ncol(x), nzv). The columns that are removed are the yellow ones in the plot, by making an image of the matrix. After the columns are removed models can be fitted. Once the pre-processing is complete, then implementing k-nearest neighbors on the MNIST dataset can be done. First, column names are added to the feature matrices because this is a requirement of the caret package.

For k-nearest neighbors, the first step is to optimize for the number of neighbors. Keep in mind that when running the algorithm, the distance between each observation in the test set and each observation in the training set will be computed. These are a lot of calculations and will therefore use k-fold cross-validation to improve speed. The caret package can be used to optimize the k-nearest neighbor algorithm. The following code, control <- trainContol(method = "cv", number = 10, p = 0.9) and train_knn <- train(x[,col_index], y, method = "knn", tuneGrid = data.frame(k = c(1,3,5,7)), trControl = control), will find the model that maximizes the accuracy. The plot below shows which number of neighbors is optimal to use. In this case, k = 3. To prevent overtraining k = 1 is not used.



Once optimizing the algorithm is done, then the entire dataset can be fitted. The code would look like this: fit_knn <- knn3(x[,col_index], y, k = 3) . The accuracy is approximately 0.97. From the specificity and sensitivity output coming from the confusion matrix function, the number nine is the hardest to detect, and the most commonly incorrect predicted digit is five. The number nine has the lowest sensitivity and number five has the lowest specificity. This can be seen using this code: cm$byClass[, 1:2].

The last step is to apply the algorithm that was optimized using the training set to test set. The knn3( ) function is used to fit the model for the test set. The accuracy is approximately 0.96.From the specificity and sensitivity output coming from the confusion matrix function, the number eight is the hardest to detect, and the most commonly incorrect predicted digit is one. The number eight has the lowest sensitivity and number one has the lowest specificity. This can be seen using this code: cm_test$byClass[, 1:2].

III. Results

In this section the results are shown. First, the accuracy of the training set is displayed. Second, the specificity and sensitivity output coming from the confusion matrix function for the training set is displayed.

Training set accuracy

```
cm$overall["Accuracy"]
```

```
## Accuracy
##     0.97
```

Training set specificity and sensitivity output

```
cm$byClass[, 1:2]
```

```
##          Sensitivity Specificity
## Class: 0   1.0000000   0.9972558
## Class: 1   0.9905660   0.9960850
## Class: 2   0.9558824   0.9977728
## Class: 3   0.9500000   0.9972222
## Class: 4   0.9698492   0.9983343
## Class: 5   0.9767442   0.9961707
## Class: 6   0.9855072   0.9988846
## Class: 7   0.9955556   0.9915493
## Class: 8   0.9278351   0.9988926
## Class: 9   0.9473684   0.9944165
```

Third, the accuracy of the test set is displayed. Fourth, the specificity and sensitivity output coming from the confusion matrix function for the test set is displayed.

Test set accuracy

```
cm_test$overall["Accuracy"]
```

```
## Accuracy
##   0.9565
```

Test set specificity and sensitivity output

```
cm_test$byClass[, 1:2]
```

```
##          Sensitivity Specificity
## Class: 0   0.9814815   0.9977578
## Class: 1   1.0000000   0.9915398
## Class: 2   0.9554455   0.9949944
## Class: 3   0.9644670   0.9927898
## Class: 4   0.9425287   0.9978094
## Class: 5   0.9301075   0.9955899
## Class: 6   0.9848485   0.9966704
## Class: 7   0.9729730   0.9906336
## Class: 8   0.8871795   0.9994460
## Class: 9   0.9363636   0.9943820
```

The accuracy values are very high. In addition, the specificity and sensitivity output values are high overall as well. The k-nearest neighbors algorithm is better than logistic regression because the values are closer to the true conditional probability.

IV. Conclusion

The goal of this project is getting the highest accuracy possible in processing the images of digits from the MNIST dataset using the k-nearest neighbors algorithm. A training dataset was used to develop the k-nearest neighbors algorithm and then it was applied to the test dataset. To reduce the time the algorithm takes to run k-fold cross-validation was used. In the k-nearest neighbors algorithm overtraining and oversmoothing is prevented by setting

k = 3. To determine the best model the accuracy parameter is measured. The approximate accuracy of 0.97 from the training set and 0.96 from the test set show that the k-nearest neighbors algorithm is very reliable for image processing.