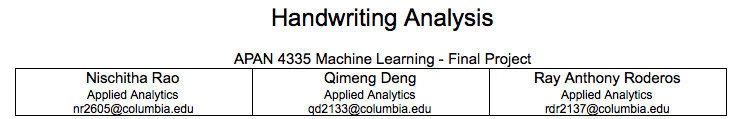
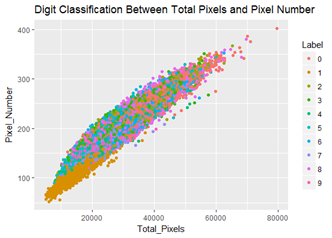
****

**Abstract**

The topic our team has chosen as our final project is Handwriting Recognition: We were able to achieve a 97.3% prediction accuracy of the MNIST test data using *h2o’s* Neural Net model with 2 layers and 600 neurons. Using this model, we accomplished an FOM of 10% with the training dataset size of 3000. We were able to achieve this through value parameter exploration.

**Introduction:** Dataset, Features and Software

The dataset used is the MNIST database of handwritten digits. Initially an MNIST excel file converted into CSV. It has a training set of 60,000 examples, and a test set of 10,000 examples. It has 785 features - 1 classifier variable categorized between 0 to 9 and 784 predictor variables. The predictor variables are pixels with values from an 8-bit grayscale (0-256) describing the intensity of the shade of gray (from white to black). If the predictors are arranged in a 28x28 matrix and plotted according to their grayscale value, an image of the handwritten digit will be created. The software used to analyze the data is the latest version of R and RStudio as of March 2017.



Classification Plot between Total Value of Pixels and Number of Pixels with Non-Zero Values

**Methodology**

Methods followed in chronological order:

1. Finding the best Neural Net Library for the dataset
2. Determined the best number of hidden layers
3. Determined the best number of neurons for each layer
4. Determined the lowest possible training set size that results into the best FOM

**Finding the Best Neural Net Library**

We began with finding the best package to use the neural net for image recognition. Based on research, we tried the “neuralnet”, “nnet” and “h20” packages on the handwriting analysis. The following are brief descriptions of each package:

*neuralnet package:* this package can train neural networks using the backpropagation, resilient backpropagation with (Riedmiller, 1994) or without weight backtracking (Riedmiller, 1993) or the modified globally convergent version by Anastasiadis et al. (2005). [3]

*nnet package:*  it is for feed-forward neural networks with a single hidden layer, and for multinomial log-linear models. [4]

*h2o* package: is an open-source in-memory prediction engine. It is an optimized Java Virtual Machine for processing of distributed, parallel ML algorithms on clusters (such as a laptop).[5]

Before running the models, we created metrics that would be stored and tabulated for easier and more convenient comparison. To choose a package that would suit our model and parameters, the following metrics were analyzed: 1) Size of the training set 2) Accuracy 3) Error Rate 4) Figure of Merit (FOM) and 5) Operating Time. Based on the benchmark determined by us, at 1 layer of 100 neurons and 2 layers of 100 neurons each, the *h20* package performed much better with an accuracy above 90% while the *neuralnet* and *nnet* packages performed at only 10% accuracy. In terms of time computation, one run with the *neuralnet* package took 90 minutes while the *h20* package performed in 20 minutes and the *nnet* package ran for 10 minutes. Regarding both the accuracy and the data processing time, we moved forward with using the *h2o* package for the digit recognition.

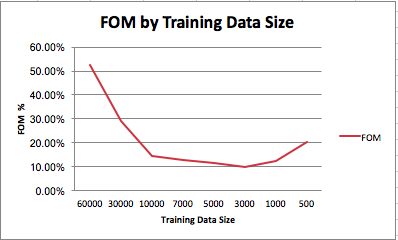
**Check Ideal Number of Layers**

With all the efforts now focused on the neural network model using the *h2o* package, our team explored on changing parameters in the *h2o* training model to find out the ideal number of layers and neurons. The methodology remained the same. We used a for-loop function running each model numerous times with the value of the chosen parameters changing for each iteration. Using too few or too many neurons in a layer may lead to overfitting or underfitting. As we had 784 features, we began to train our deep learning model from 1 layer, 100 neurons per layer, and 20 epochs (the default value was 10, but we decided to stream the dataset 20 times to accommodate the size of the dataset). We changed the number of layers from 1 to 8, but the accuracy revolved around 96.5% and the FOM was around 0.53. As the model accuracy between using 2 layers and using 8 layers was not significantly different but the training time became tediously longer after adding to 5 layers, we decided that 2 would be the ideal number of layers.

**Parameter Tuning**

Starting from 100 neurons per layer, we added 100 more neurons per layer in each new training model and stopped at 600 neurons per layer. When neurons per layer were between 200 and 600, we found that the accuracy would be around 96.5% or slightly higher, the FOM would be around 0.525 and the changes in accuracy were minimal. The highest accuracy we achieved was 97.3%, which was generated by a 2-layer, 600-neuron and 20-epoch model. So we increased the epochs to see if we could get better results.

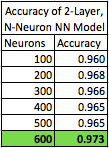
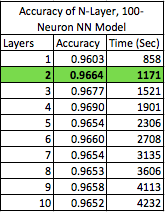
When we used 2 layers, 600 neurons per layer and 30 epochs, we found that the accuracy did not change compared to 20 epochs. So we decided to maintain 20 epochs and tried to reduce the training dataset size to reduce the FOM.



**Reducing FOM – Bootstrapping**

With the parameters of the Neural Net Model now optimized with an accuracy above 97%. Our team then focused on decreasing FOM as much as possible by making the training set as small as possible. To be able to quickly gauge the lowest possible FOM, we employed another for-loop function with decreasing size of the training set by a factor of 10,000 (60k, 50k,40k,30k,20k, 10k) and after 10k, by a factor of 1000. The results showed that the lowest possible FOM achieved was at 3000 observations with an FOM of 10%. For each iteration that the training set was reduced, we bootstrapped the reduced training set to its original 60k size. The method performed was a random resampling from the reduced training set until it was 60,000.

**Ideal Number of Layers Parameter Tuning**



**Results**

The final results, parameters and training size of the Neural Net Model for Image Recognition are the following:

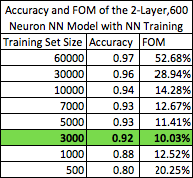
Model: *h2o’s* Deep Neural Net with 2 layers, 600 neurons each with a tanh activation function at 20 iterations

Training Set Size: 3,000 (5%)

Accuracy: 90.3%

Error Rate: 10.7%

FOM: 10%



**Learning and Recommendations**

One of our learnings during this project has been the complexity of the model. Training a Neural Net with a large data set on our Computers was a very taxing and tedious computation process. We recommend exploring other programming languages such as Python to process the dataset and compare the prediction result using different models and methods. It may also contain packages that optimize the deep learning algorithm. We also recommend using a 2nd or more powerful computer to train the complex model. We also learned the importance of having a workflow of the training and prediction trials of the model since it took a very long time. We learned how to partition smaller data as a test to check if our code is correct and to prototype our model. Once we determined that the code created, matched our criteria, we then trained and validated several models using the test set with the computer overnight. Since we used various classification models for the MNIST dataset in our midterm project, we were able to compare the results of the classical ML methods with the Neural Net. We learned that NN performed much better than other classical methods with almost no feature engineering at almost the same operational time.

**Conclusion:**

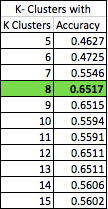
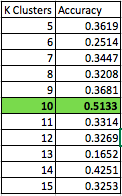
Combining our experiences from using the MNIST dataset in the mid-term project, we can conclude that the Neural Network is a very powerful tool for image recognition and can perform well with a relatively smaller dataset and is more effective than the classical ML approaches.

**Additional Objectives**

Accuracy of each cluster for recognizing the corresponding labels:

1. Clustering with Normal Features: We used an unsupervised model of K-means Cluster Analysis with parameters k= 5:15, to test the optimal accuracy. We then labelled each cluster to the digit that had the highest frequency in that cluster. Naturally, there will be clusters with identical digits and some digits without any clusters. To make sure that the missing digits were included in the clusters, we created an additional label. This resulted in some of the clusters being labeled with more than one digit. We then created a confusion matrix and computed the results. We arrived at a conclusion that k=5 was most accurate. Below is the accuracy table for all the parameters that was tested for all possible values of k= 5:15
2. Clustering with Hidden Layer Outputs:

Our Team retrieved the information from the 1st layer output and redid the exercises. We redid the unsupervised learning model KNN with K clusters. We were able to achieve the clustering at 8 clusters with 65.7% accuracy.

1.  (II)

**Sources:**

[1] Victoria Catterson. (2013, December 23). Understanding data science: clustering with k-means in R. Retrieved from <http://cowlet.org/2013/12/23/understanding-data-science-clustering-with-k-means-in-r.html>

[2] Frauke Günther, and Stefan Fritsch. “Neuralnet: Training of Neural Networks.” RJournal 2 (2010): 30–38. Print. Retrieved from <https://journal.r-project.org/archive/2010-1/RJournal_2010-1_Guenther+Fritsch.pdf>

[3] Stefan Fritsch, Frauke Guenther, Marc Suling, Sebastian M. Mueller. Training of Neural Networks. 2016, Aug 16.

<https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf>

[4] Brian Ripley, William Venables. Feed-Forward Neural Networks and Multinomial Log-Linear Models. 2016, Feb 02. Retrived from <https://cran.r-project.org/web/packages/nnet/nnet.pdf>

[5] The H2O.ai team. R Interface for H2O. 2017, Apr 15. Retrieved from <https://cran.r-project.org/web/packages/h2o/h2o.pdf>

[6] Joseph Rickert. Diving into H2O. 2014, Apr 17. Retrieved from <http://blog.revolutionanalytics.com/2014/04/a-dive-into-h2o.html>

**Supplementary Materials:**

1. R- code File