Technical Report: Digit Recognition

Group 6: Yilin Li, Hien Nguyen, Lyn Peterson

12/6/2019

Abstract

A brief overview of the area that you'll be investigating, the research question(s) of interest, your approach to analysis, and the general conclusions.

Overview:

Research question: Can we build classifiers to recognize what the digit (0-9) is in a given image based on 60,000 training images?

Approach to analysis:

General conclusions:

Introduction

The MNIST database of handwritten digits is a classic dataset for machine learners. Most existing models nowadays use different geometric properties of digits from complex feature selection. The best classifcation rate is produced by convolutional neural nets, which is 99.77%. We woud like to build classifiers, using our knowledge from this course, to recognize what digit (0-9) is displayed in a given image. We are interested in seeing how our models would perform with respect to literature results.

The Data

The data comes from the MNIST database of handwritten digits displayed in 28x28 greyscale images. The digits have been size-normalized and algorithmically centered in a fixed-size image.

The images are split into two different subsets, a training set containing 60,000 images and a test set containing 10,000 images. The two subsets are completely disjoint.

Data processing Since the pre-downloaded data is already well-formatted, minimal effort was spent on data processing. We performed two additional steps of data processing:

(1) Each 28x28 image was flattened into a single row of 784 pixels.

(2) All pixels were rescaled from 0-255 to 0-1.

Exploratory Data Analysis

A closer look at the data

As shown by the frequency graph (Figure 1), each digit has a roughly equal amount of input samples. This suggests that the response classes are well separated and evenly distributed.

We can see that the digits are centered and manipulated with some anti-aliasing technique (Figure 2). The dataset seems to encompass a variety of handwritting styles of digits. For example, among the six 1's images in Figure 2, different styles are displayed such as italic and formal.

For each digit, we averaged over all pixels and obtained a mean image (Figure 3).

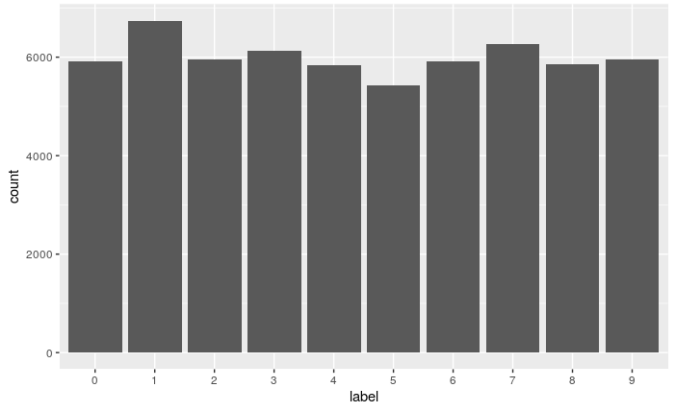

Figure 1: A bar chart of frequency of input samples for each class of digits.

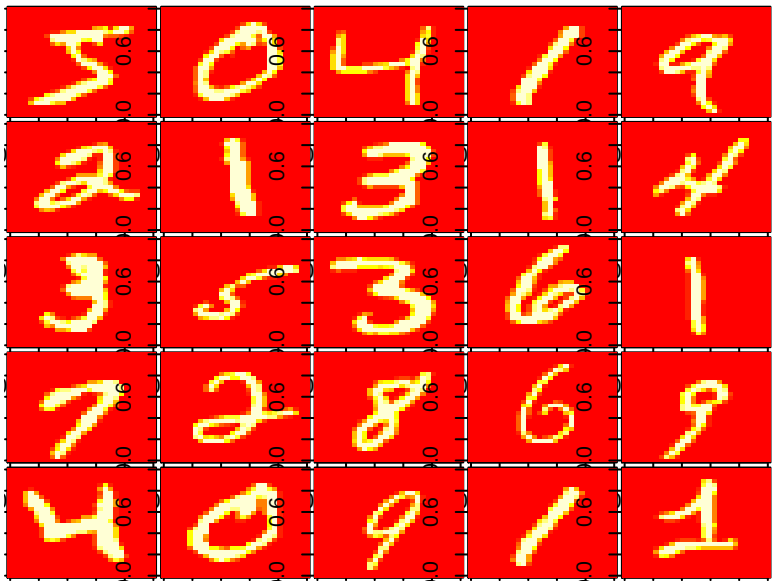

Figure 2: First 25 images in the training dataset

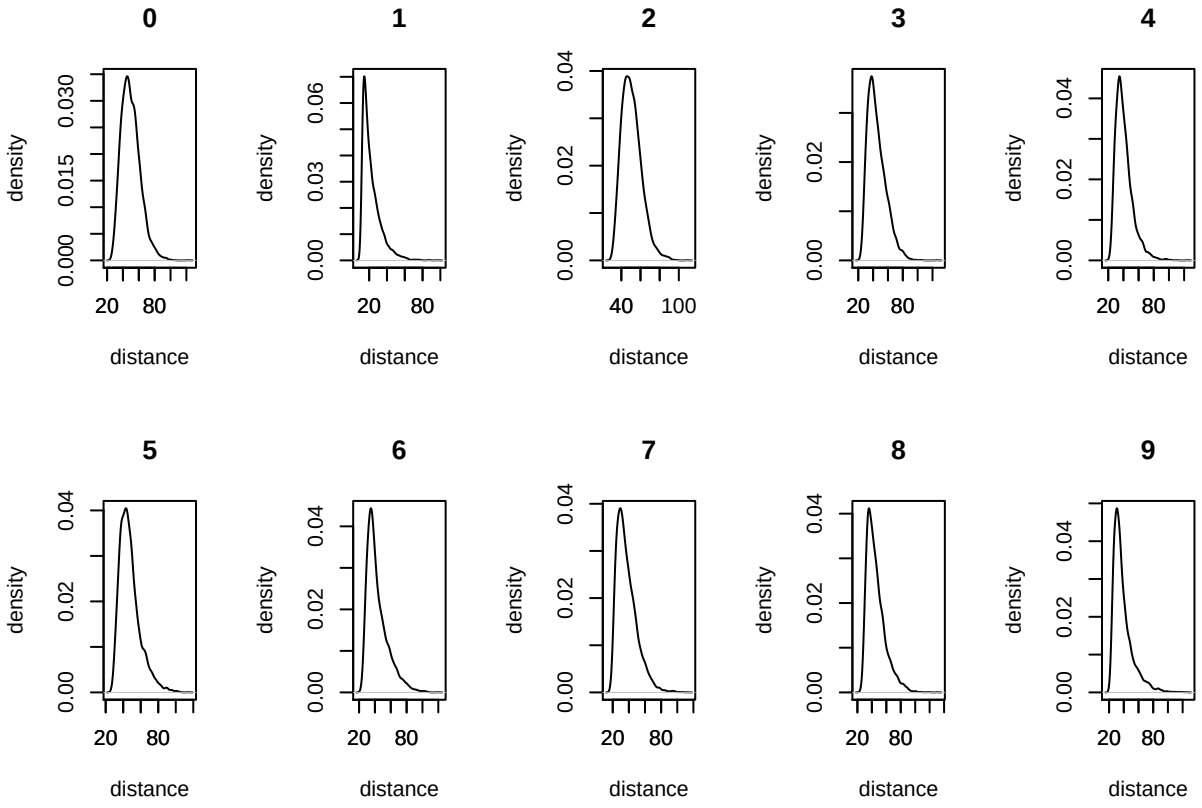

Figure 3: Density distribution of the Euclidean distances from each input image to its mean form

We computed the Euclidean distance from each input to its mean image. We plot the image-to-mean distance distribution for each digit.

From the distance summary table (Appendix A), we can see that digit 1 has the lowest mean distance; this indicates that most people write it similarly. 2 has the highest mean distance, so people tend to write 2 in different ways. We also see that 6 and 9 have the highest distance variance, which suggests these two digits have the most variation in people's writing styles.

Principal Component Analysis & Early Predictions

Since the MNIST dataset is large, we employed an unsupervised technique, Principal Component Analysis (PCA), to reduce the dimension of the dataset. We fit PCA on both the training and test dataset. From the scree plot (Figure 5b), we found that the first 20 principal components were able to capture around 90% of the variance in our data.

From the PCA plot of the first two principal components (Figure 5a), we observed that the 1-0 pair is particularly well-separated, whereas there is considerable overlapping between 4 and 9. Based on the distribution of digits, we hypothesized that PC1 represents the denseness of central pixels for each digit. For example, an image displaying a "0" will have more pixels in the outer area (of a 28x28 box), while an image of a "1" will see more pixels aligned in the middle. Thus, the PC1 score for 0's will be greater than 1's, which explains the clear separation of the two classes across the PC1 axis.

We also hypothesized that PC2 is related to horizontal symmetry. For example, 3 and 1 are symmetric horizontally, which have larger PC2; 9 and 7 are less horizontally symmetric, which have lower PC2.

Based on the PC graph above, we predicted that the 1-0 pair would be easily recognizable and 4-9 might be harder to distinguish.

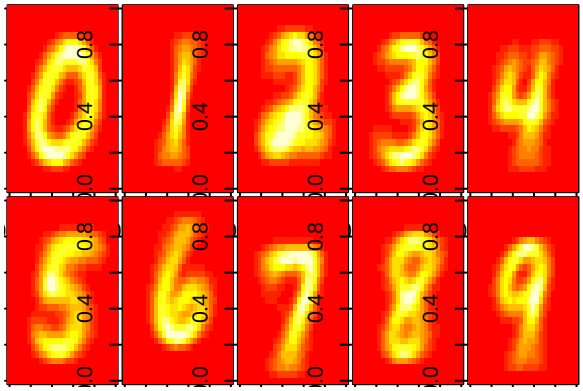

Figure 4: Mean image for each digit

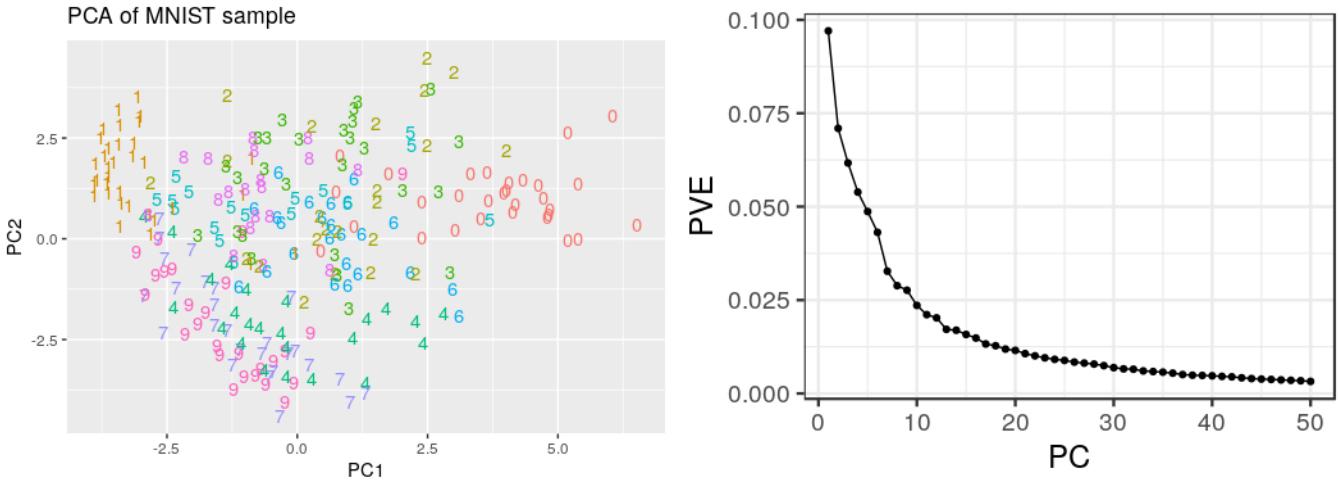

Figure 5: (a). PCA plot of the first two principal components. (b) Scree plot of the first 50 PCs.

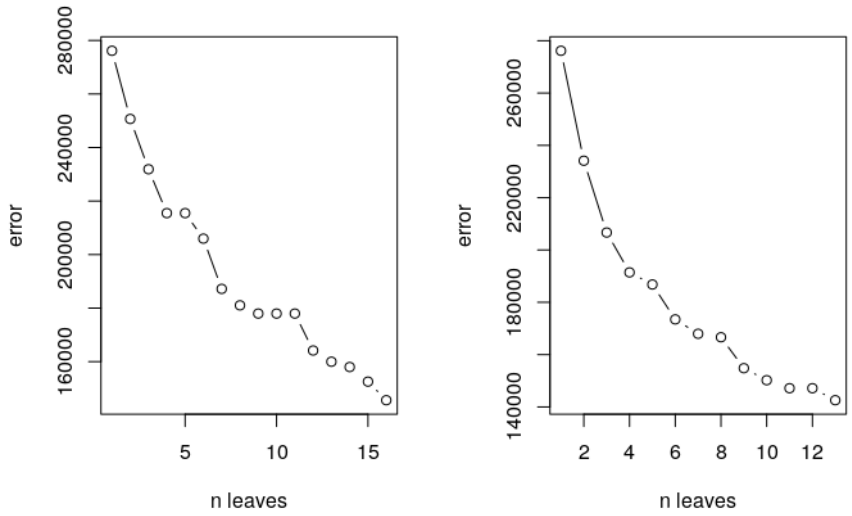

Figure 6: Pruning tree on both datasets (a) original dataset, and (b) PCA-reduced dataset.

Modeling

We performed modelling on both the original training dataset (60,000 samples with 784 predictors) and the dimension-reduced dataset using PCA (60,000 samples with 20 predictors). We experimented with both parametric (Logistic Regression) and non-parametric models (Trees, KNN, SVM). All of our models are cross-validated. We compared the model performance on both datasets as well as with published results, and summarized the resutls (Table 1).

For each model, we looked at the confusion matrix to find pairs that were particularly interesting (easy or difficult to predict). We measured the model accurary based on the misclassification rate, that is, the percent of wrong predictions made by the model.

- **Classification Tree**

Our first pruned classification tree had limited success. Even without pruning, the optimal number of nodes was selected (n = 16) (Figure 6a), and the graph of error against size showed no clear elbow indicating optimal size.

Looking at the tree plot (Figure 7), the important pixels (at each split) seem to be near the center of the image (around 400). The misclassification rate is near 40% (36.52%), as this is a rather weak model with a limited numer of splits.

The first tree could not classify the class 5 with 16 splits (using the `tree` package), so we created a similar tree using another package (`rpart`) that successfully modeled all classes with only 14 splits (Appendix B). However, the misclassification rate for this model is slightly higher than the previous tree (38.04%), perhaps due to having fewer splits.

On the PCA-reduced dataset, the best tree size was, again, naturally chosen (n = 13) by the model (Figure 6b).

From the tree plot (Figure 8), we can see that the two biggest splits happen at PC2 and PC1. These splits divided the PCA plot with two principal components into four regions. We observed that the digit 1 nearly occupies the entirety of one region, which led to our hypothesis that this would be the easiest class to classify. We also noticed two groupings of digits, 8-5-3-0 in the upper region and 4-9-7-6 in the lower region. We expected the misclassification rate to be higher for these digits since they are closer to one another (Figure 9).

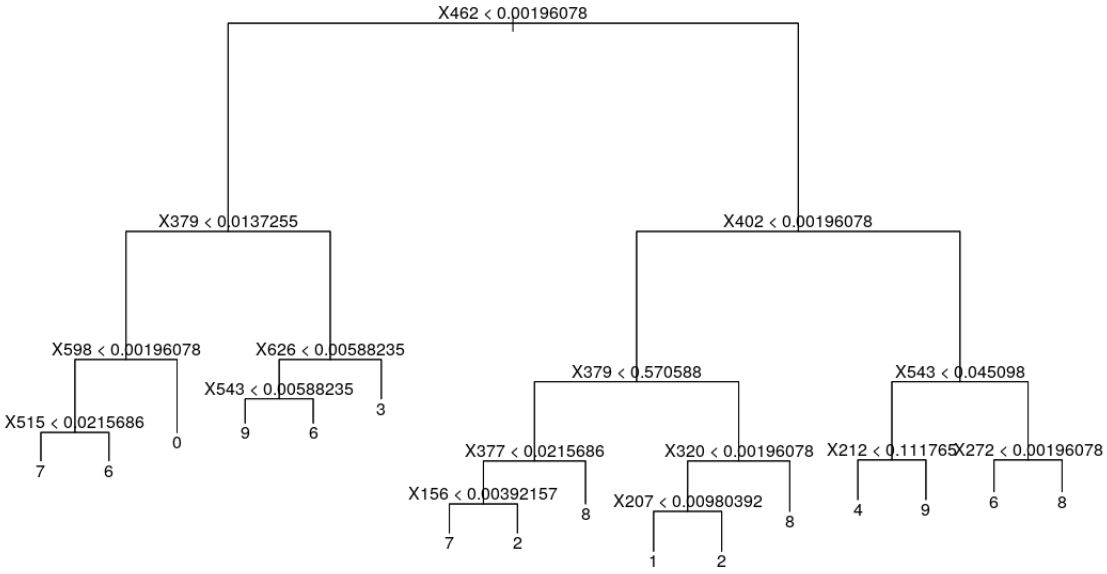

Figure 7: Tree plot for the original dataset

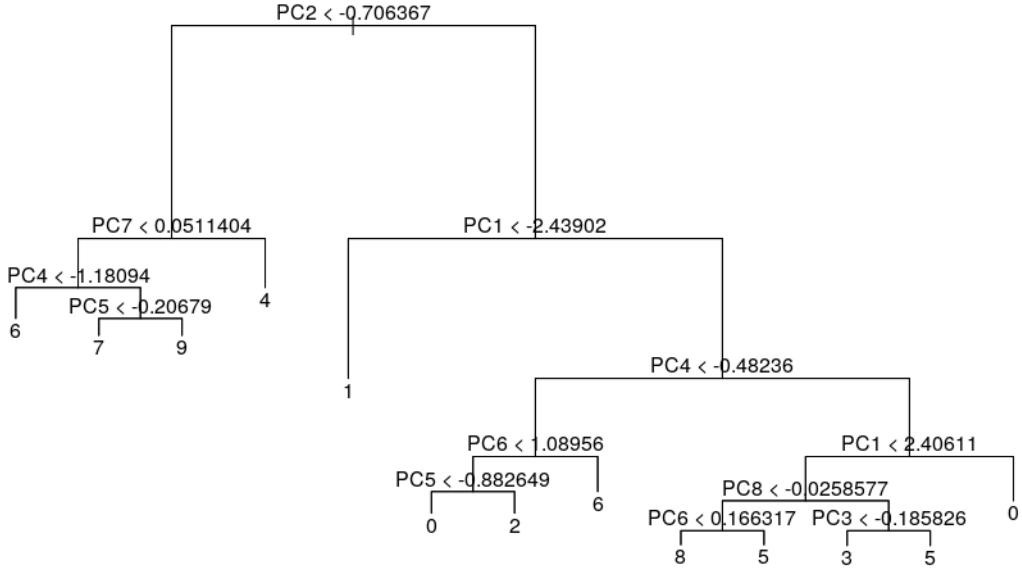

Figure 8: Tree plot for the PCA-reduced dataset.

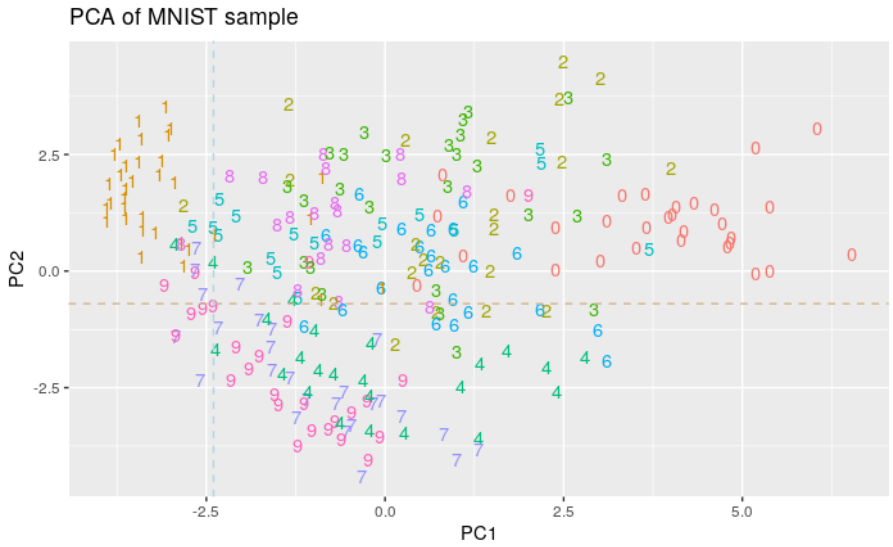

Figure 9: PCA plot of the first two PCs revisited. The dashed lines represent the tree splits along the PC1 and PC2 axes. Blue line separates the classes into left and right regions. Brown line separates the classes into upper and lower regions.

The misclassification rate for the tree model on the PCA-reduced dataset closely mirrors the result on the original dataset at 36.43%.

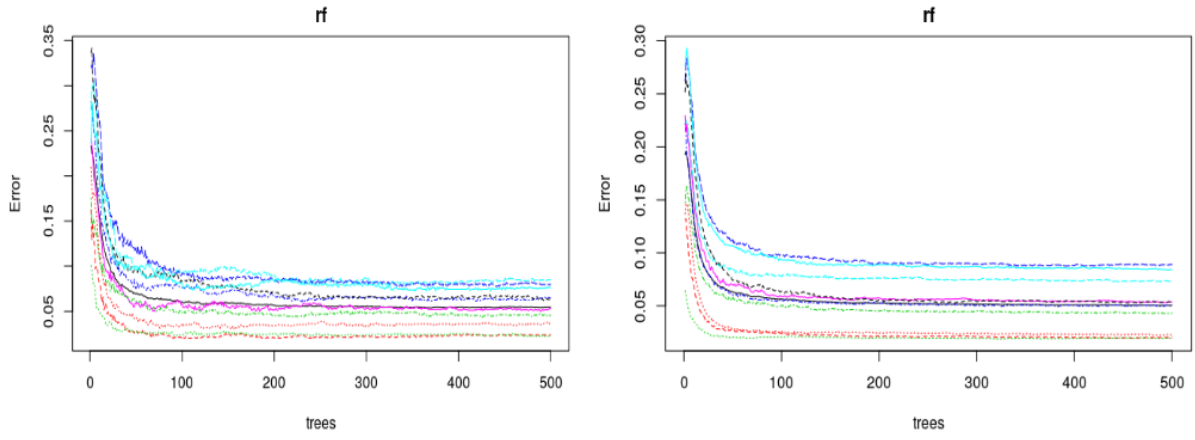

Figure 10: Choosing size of the random forest on (a) original dataset and (b) PCA-reduced dataset. Each colored line represents the error rate of a class of digits 0-9. Errors flatline for any number of trees greater than 50 on both datasets.

- **Random Forest**

We chose $m = p/3$ for the random forest.

For the original dataset, the plot of error against size (Figure 10a) shows that as long as there are 50-100 trees, error is rather low.

The mean decrease accuracy chart (Figure 11) shows that the random forest identified 7-13 most important pixels, as did the simple tree.

For the PCA-reduced dataset, we also observed that the error rates for all classes gradually flatline when the best number of trees exceeds 50. The random forest considered all 20 PCs to be important predictors, as none of which was scored 0 in both the mean decrease accuracy and the mean decrease gini plots. The random forest identified 7-9 principal components as more important than others, among which PC1, PC2 and PC4 are considered the most important (Figure 12).

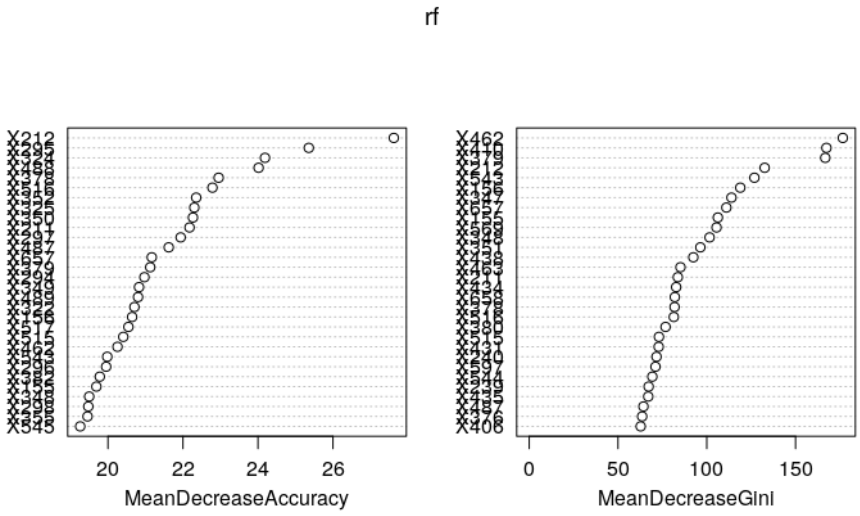

Figure 11: Variable importance plot by accuracy score and GINI index on the original dataset for the random forest.

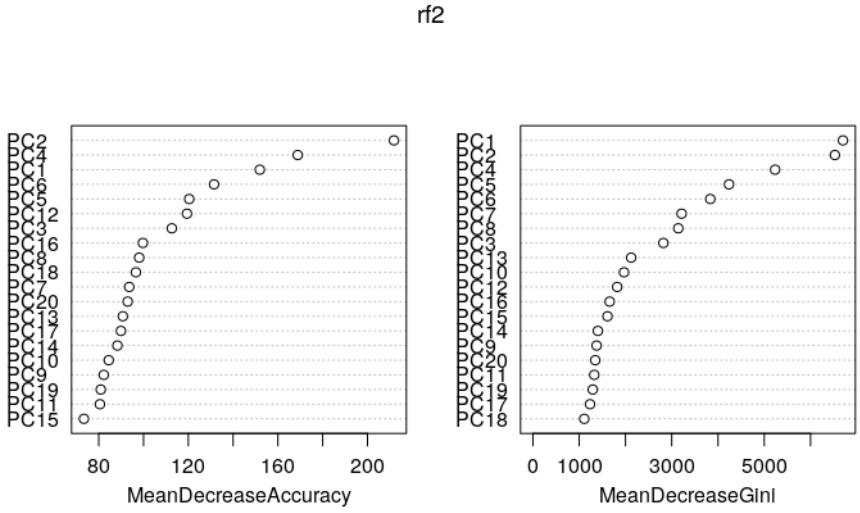

Figure 12: Variable importance plot by accuracy score and GINI index on the PCA-reduced dataset for the random forest.

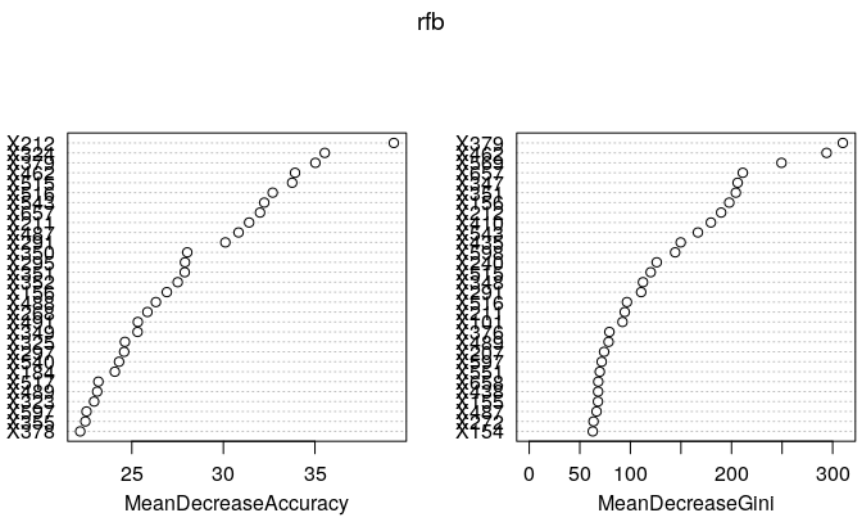

Figure 13: Variable importance plot by accuracy score and GINI index on the original dataset for the bagged trees.

- **Bagged Trees**

On the original dataset, the bagged trees ($m = p$) also identified a similar number of important pixels that seem to be near the center of the image. The error rate was the lowest when m=1 and the highest when $m = p$, suggesting that there might be some overfitting in the random forest, as it fairs better when fewer predictors are considered at each split.

On the PCA-reduced dataset, the bagged trees also consider all 20 PCs important predictors, among which PC1, PC2, PC4 were the top ranked predictors.

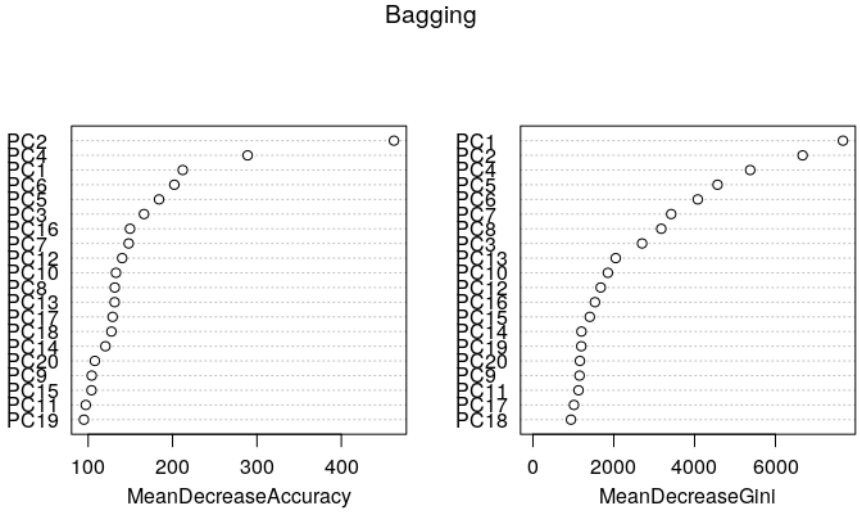

Figure 14: Variable importance plot by accuracy score and GINI index on the PCA-reduced dataset for the bagged trees.

- **Boosted Trees**

We cross-validated the boosted model on both datasets with 5-fold CV to pick the best number of trees. As a result, 454 trees were chosen for the original dataset and 411 for the PCA-reduced dataset.

On the original dataset, 560 out of 784 predictors had non-zero influence, as scored by the boosted model (Figure 15a).

On the PCA-reduced dataset, the boosted model considered 19 out of 20 PCs to be statistically significant. The variable importance table shows that PC1, PC2 and PC4 were again the top predictors (Figure 15b).

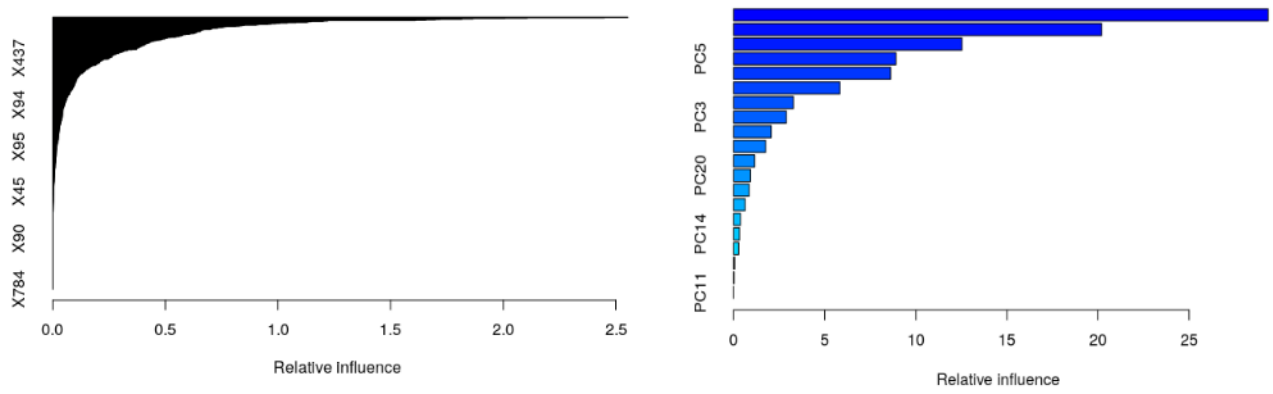

Figure 15: Relative influence plot by boosting on the (a) original dataset, and (b) PCA-reduced dataset.

- **Multinomial Logistic Regression (with Ridge and Lasso)**

We present two approaches to perform Multinomial Logistic Regression.

(a) Using one-vs-all classification:

We trained 10 different (binary) logistic regression classifiers, each recognizing one class of digits. For both the training and test dataset, we replaced the initial response variable (a nomial variable with 10 levels) with 10 dummy variables corresponding to the digits 0-9, each taking a value of (1) if the true label for the current digit was given, and (0) otherwise. We then assigned the new response variable to each of the classifier, and performed training using logistic regression with the `glm()` function.

Using the trained logistic classifiers, we computed the probability that the current digit belongs to a class. For each input, we picked the class for which the corresponding logistic regression classifier outputs the highest probability as the predicted class. We compared the returned prediction vector with the true labels to find the model accuracy. The misclassification rate using this method is 13.15%.

(b) Using built-in package:

We verified the above result using a built-in package called `nnet`, which supplies the `multinom()` function for multi-class logistic regression. The misclassification rate is 12.88%, which suggests that there is not much difference between the two methods.

To prevent overfitting, we applied Ridge and Lasso regularization.

[Caption] x-axis is λ values, y-axis is multinomial deviance.

Clearly, as λ increases, the model performs worse. The best model performance is observed at $\lambda = 0$.

Our lowest misclassification rate is 7.89% (using Ridge) on the original dataset and 11.95% (using Lasso) for PCA-reduced dataset, respectively. This further confirms that we did not encounter the problem of overfitting in our logistic model.

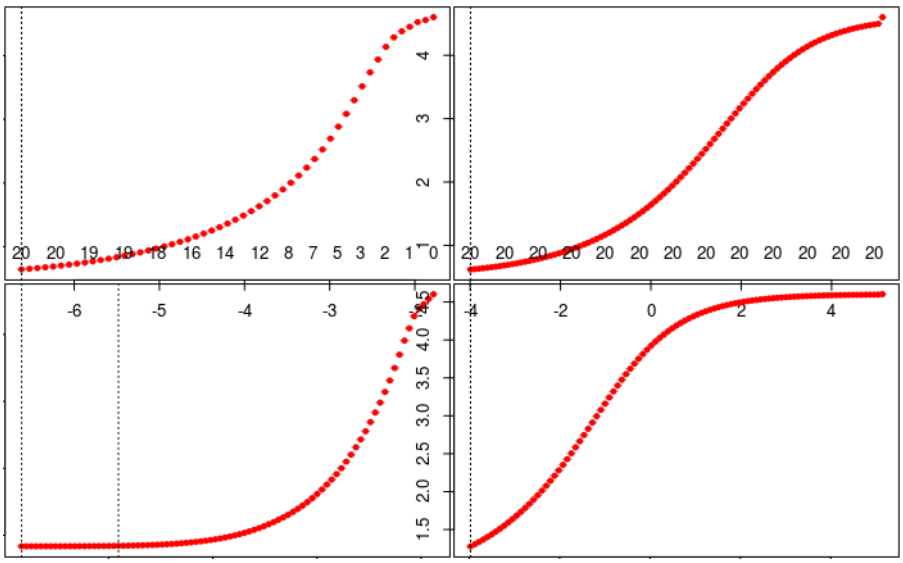

Figure 16: Ridge and Lasso regularization.

- k **Nearest Neighbors**

We performed k-nearest neighbors (KNN) on both datasets. We used leave-one-out cross-validation (LOOCV) on the PCA-reduced data set to select the best k value, which is 5.

As it took too long to perform cross-validation on the original dataset, we also applied KNN with $k = 5$ for this dataset. As a non-parametric model, KNN generates the second best result among all of our models. The misclassification rate is 3.06% on the original dataset and 3.02% on the PCA-reduced dataset respectively.

- **Support Vector Machine**

One of the most computationally intensive models we experimented with in this project is Support Vector Machine (SVM).

In the simplest sense, SVM finds a hyperplane that best separates two different classes. Using a kernel trick, SVM maps non-linear data points to a higher-dimensional space with arbitrary complexity. The model can then find a hyperplane that separates the samples in the transformed space.

We can think of this as an automated way of creating polynomial terms or interaction terms in logistic regression. We chose the "radial" kernel (RBF or Gaussian) and achieved a **2.17%** test error rate, which is our best result so far. Below is the plot of the class regions for the SVM model using the first two PCs as the predictors.

The built-in `plot()` method doesn't give us a clear visualization because we have 10 classes and 20 predictors, but it still shows us some decision boundaries with only 2 predictors, PC1 and PC2.

Unfortunately, we lacked the computing power to apply SVM on the original dataset, so we had to give up on this model eventually. However, we are positive that with the right choosing of kernel and tuned parameters, the SVM will generate a better result than 2.17% on the original dataset.

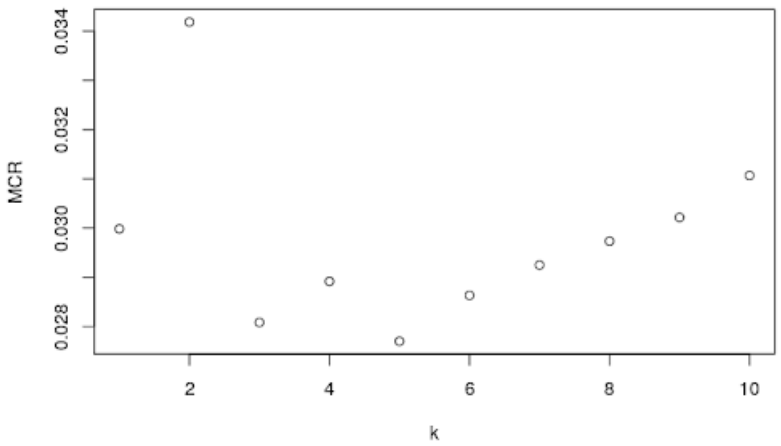

Figure 17: Choosing k value for KNN. The best k was chosen at 5.

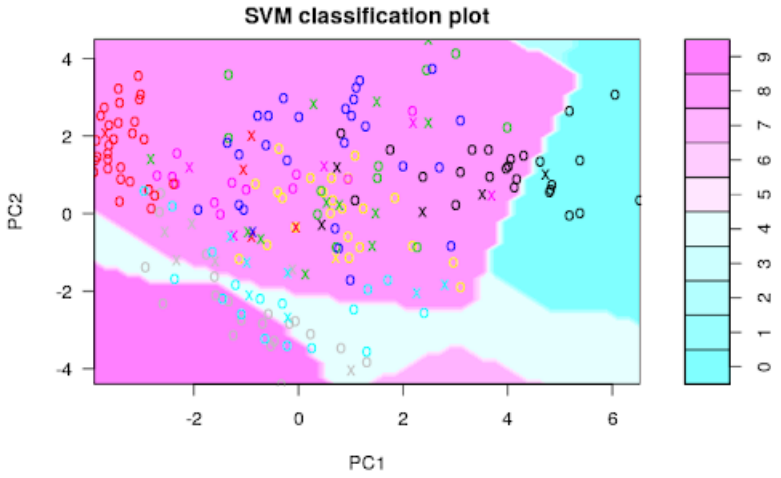

Figure 18: Visualization of SVM.

Table 1: Model summary

Model	Original dataset	PCA-reduced dataset	Published results
Pruned Tree	36.52	36.43	
Random Forest	5.05 *4.29 (m=1)	5.15	
Bagged Trees	6.05	4.89	
Boosted Trees	8.38	9.81	7.7
Logistic Regression	8.37 7.89 (Ridge) 8.29 (Lasso)	12.88 14.22 (Ridge) 11.95 (Lasso)	12
KNN	3.06	3.02	3.09
SVM	TBD	**2.17**	1.4

Discussion

Based on the confusion matrices from prediction results across models (Appendix C), we veried our assumptions that 4-9 is the most difficult pair to separate, whereas 1 was consistently the easiest class to predict. Other difficult pairs we found were 8-5, 5-3, 5-0. We found little evidence of overfitting in our models for both the original and dimension-reduced datasets, as models with greater complexity tend to perform better. We noticed some possibility of overfitting on our original dataset, particularly with the random forest and bagged trees: as the number of predictors increased, the model accuracy decreased. We did not encounter overfitting problem in the dimension-reduced dataset.

Overall, the models trained on the dimension-reduced dataset performed on par with those trained on the whole dataset, suggesting that PCA with only 20 principal components captured well most of the existing structure in our data. On the original dataset, we noticed that the center pixels were among the most significant predictors in our models, suggesting that the central area of the image contains the most information. On the reduced dataset, we also found that PC1, PC2, and PC4 were consistently the top ranked predictors across models among the selected 20 principal components. While we are not exactly sure what the principal components actually represent, we hypothesized that PC1 and PC2 capture the center alignment and horizonal symmetry of the digits, respectively.

Our best model so far is a RBF (Gaussian) kernel SVM. Compared with reported results on the website of the MNIST database, we found our models gave competitive performance with published models, which suggests that our finetuning of the parameters was effective.

Since the data had been cleaned and carefully preprocessed for us before our modelling process, future work might want to measure the effect of data preprocessing on model performance with unprocessed images. We also wish to test our claims on what the principal components might represent, perhaps with advanced visualization techniques. As we had little luck training on the more computationally intensive models on this dataset due to limited computing power at hand, we hope to overcome this limitation in the future, perhaps with more computing resources or advanced techniques.

References

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition.” Proceedings of the IEEE, 86(11):2278-2324, November 1998. Online Version.

Appendix A

```
##       mean      var
## 0 49.91889 140.5000
## 1 22.47562 107.3414
## 2 50.76020 107.3001
## 3 44.94435 150.5974
## 4 40.87232 130.9288
## 5 47.58101 142.0700
## 6 43.09167 179.6460
## 7 37.47072 157.1436
## 8 45.59575 165.0296
## 9 38.56896 179.6759
```

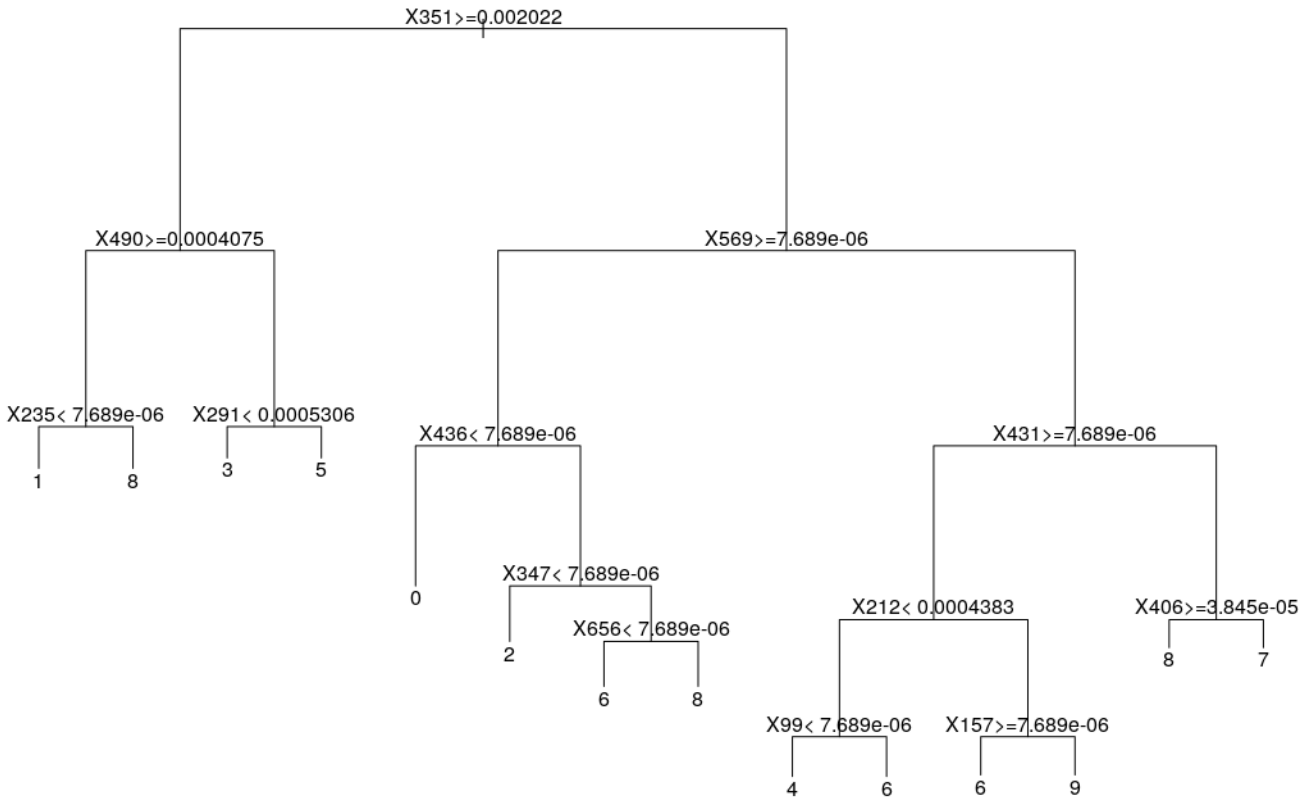

Figure 19: Tree plot on the original dataset from the 'rpart' package.

Appendix B

Appendix C

(confusion matrix for each model)