Technical Report: Digit Recognition

Group 6: Yilin Li, Hien Nguyen, Lyn Peterson

12/6/2019

Your technical report should be an .Rmd file that contains the following sections. So as not to make the compilation (knitting) of the document not take too long, consider setting cache = TRUE in the curly braces of any R chunk with substantial computing. Please knit both to pdf and github document (.md).

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

Overview of the setting of the data, existing theories/models (particularly if you are working in a descriptive/inferential setting), and your research questions.

The MNIST database of handwritten digits is a classic dataset for machine learning learners. Most existing models nowadays use different geometric properties of digits from complex feature selections. The best classifcation rate is produced by convolutional neural nets which is 99.77%. With so many well built models existing, can we build classifiers covered in this course to recognize what the digit (0-9) is in a given image and hopefully outperform several published models?

The Data

Where does the data come from? How many observations? How many variables? What does each observation refer to (what is the observational unit)? What sorts of data processing was necessary to get the data in shape for analysis?

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.

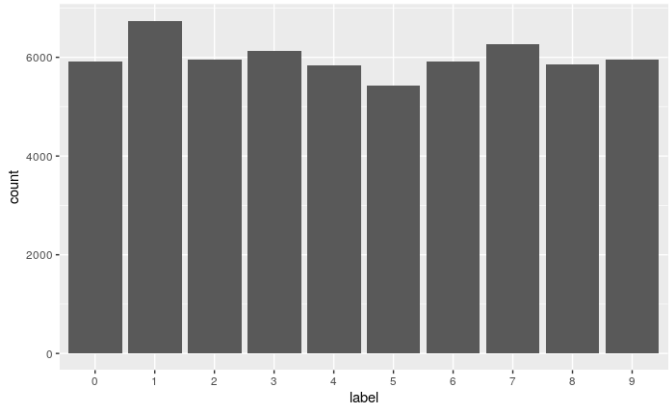

Figure 1: A bar chart of frequency of input samples for each class of digits.

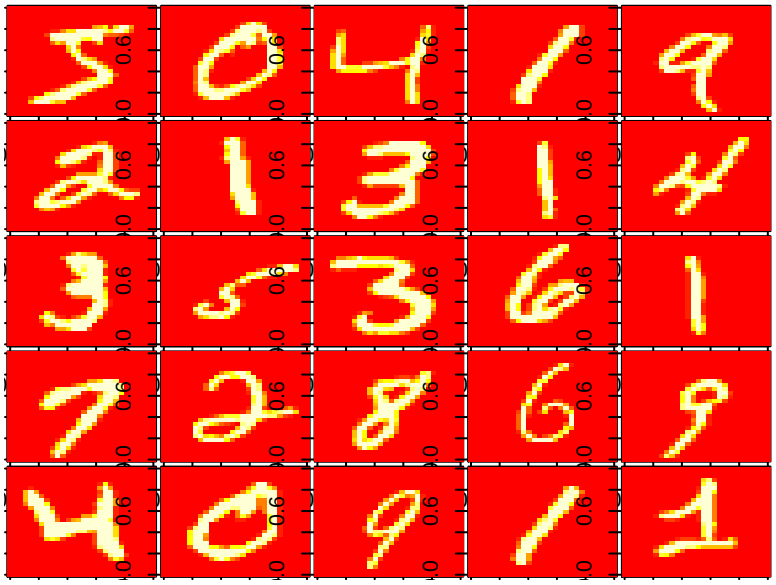

Figure 2: First 25 images in the trainig dataset

Exploratory Data Analysis

Explore the structure of the data through graphics. Here you can utilize both traditional plots as well as methods from unsupervised learning. Understanding the distribution of your response is particular important, but also investigate bivariate and higher-order relationships that you expect to be particular interesting.

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).

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

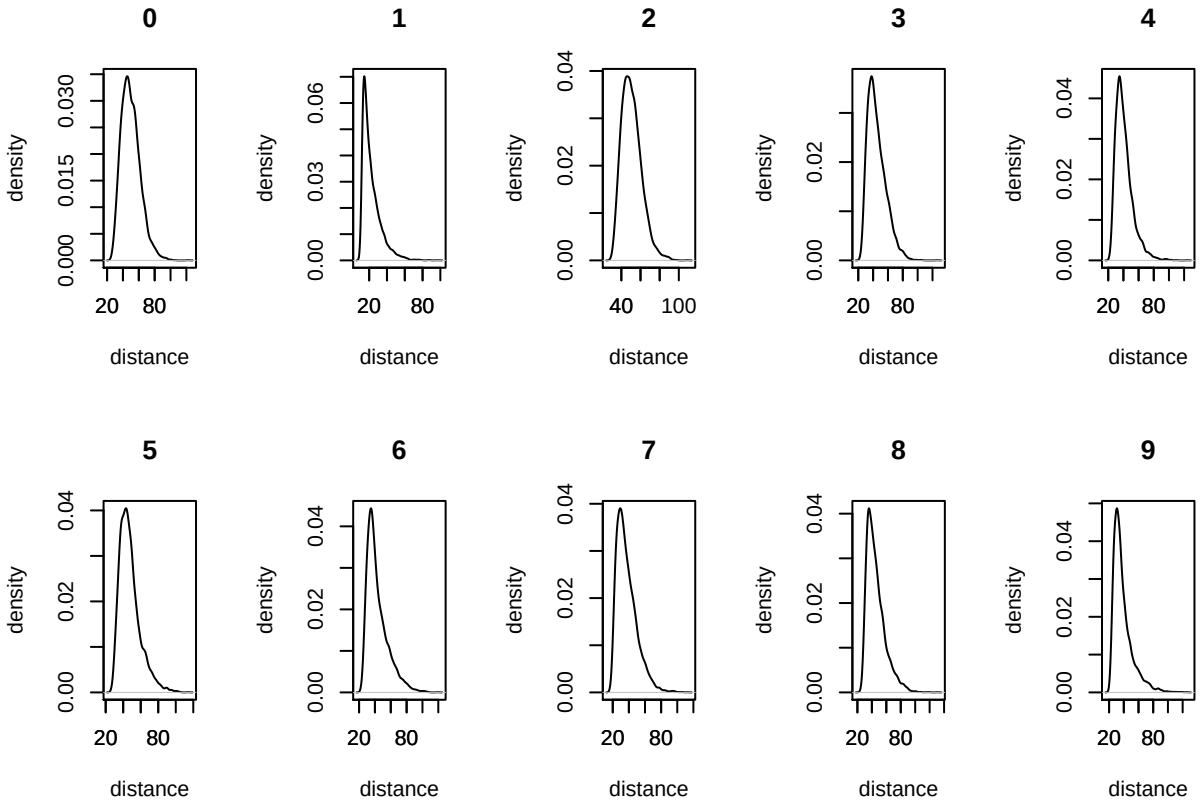

Figure 3: Density distribution of the Euclidean distances from each input image to its mean form

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 will be easily recognized and 4-9 might be hard to distinguish.

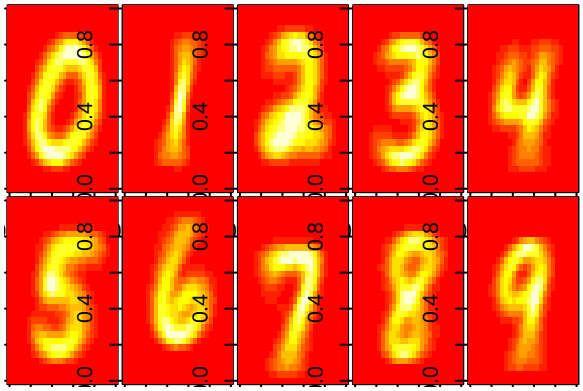

Figure 4: Mean image for each digit

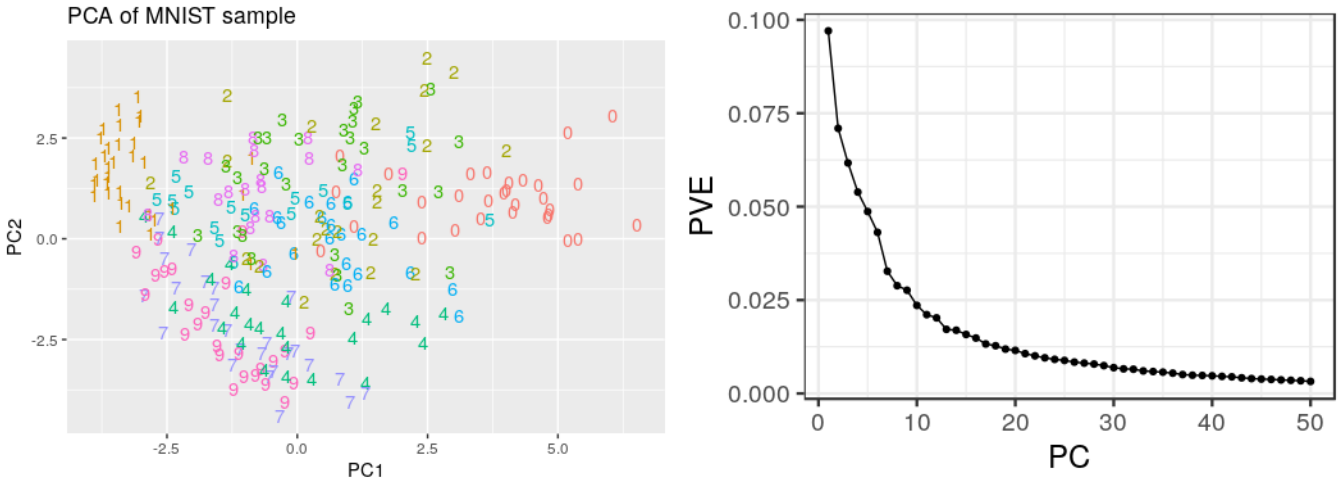

Figure 5: (a). PCA plot of the first two principal components. (b) Scree plot of the first 50 PCs.

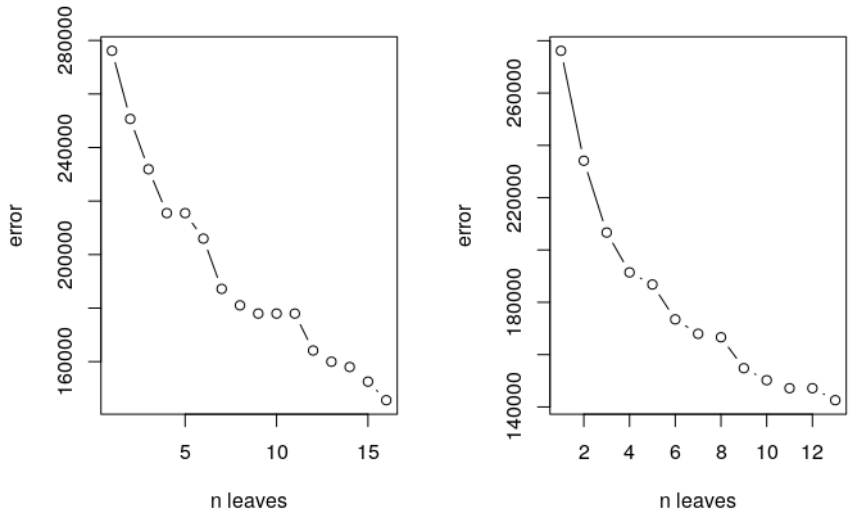

Figure 6: Pruning tree on both datasets (a) original dataset. (b) PCA-reduced dataset.

Modeling

- **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).

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

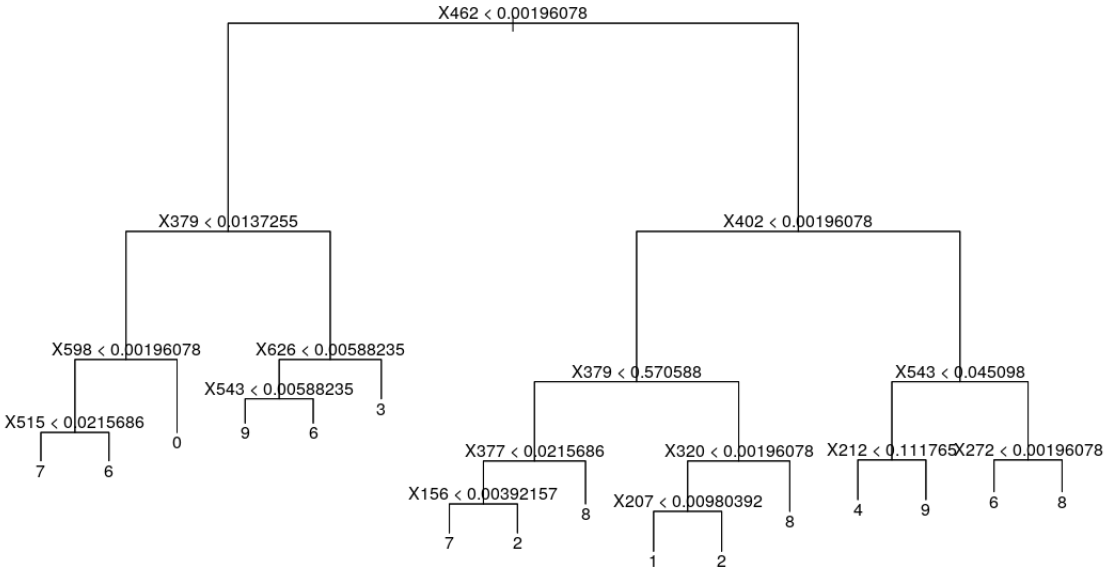

Figure 7: Tree plot for the original dataset

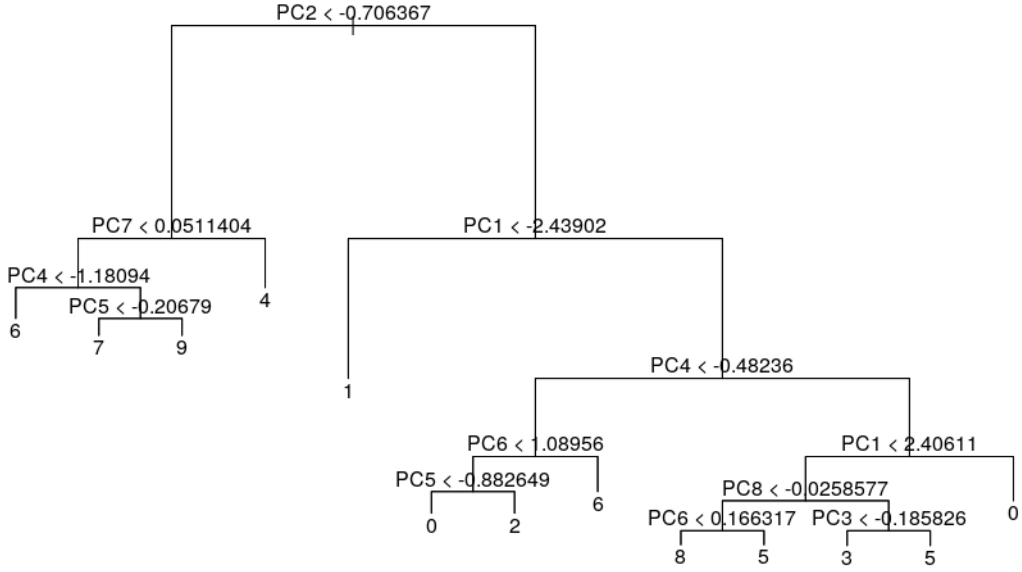

Figure 8: Tree plot for the PCA-reduced dataset.

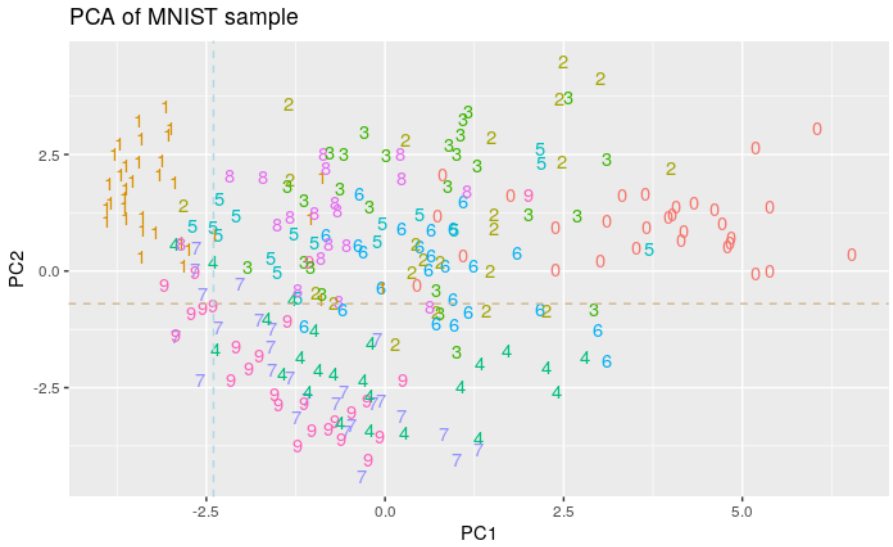

Figure 9: PCA plot of the first two PCs revisited. The dashed lines represent the tree splits along the PC1 and PC2 axes. Blue line separates the classes into left and right regions. Brown line separates the classes into upper and lower regions.

- **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).

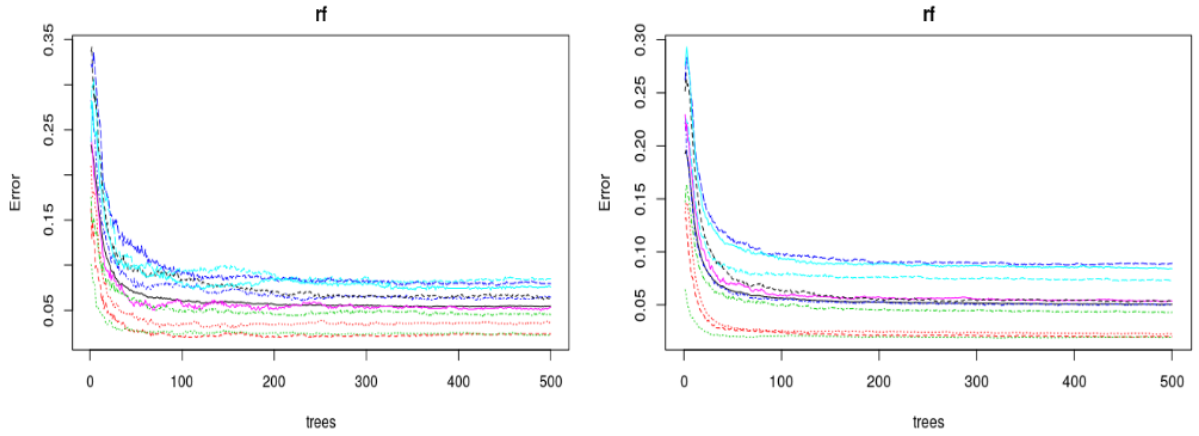

Figure 10: Choosing size of the random forest on (a) original dataset and (b) PCA-reduced dataset. Each colored line represents the error rate of a class of digits 0-9. Errors flatline for any number of trees greater than 50 on both datasets.

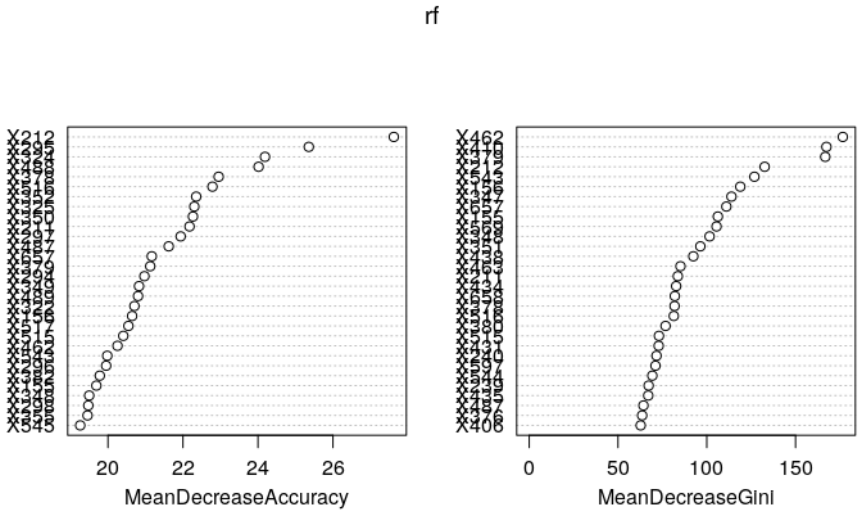

Figure 11: Variable importance plot by accuracy score and GINI index on the original dataset for the random forest.

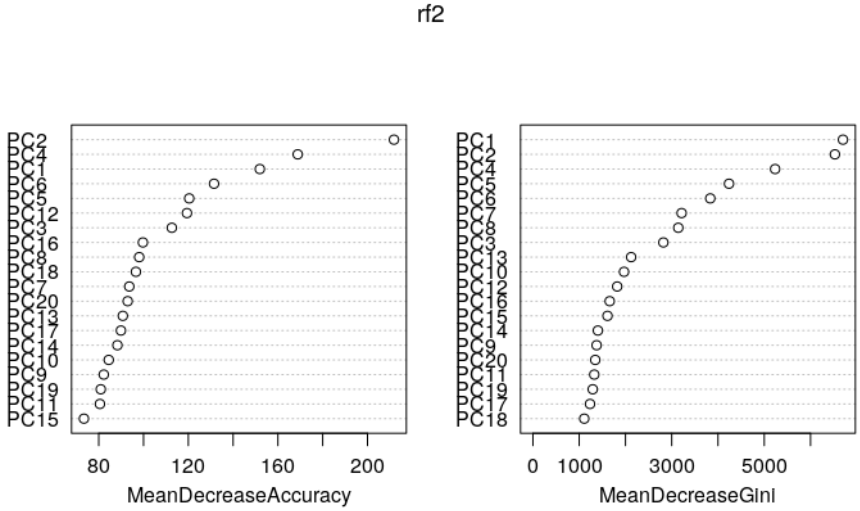

Figure 12: Variable importance plot by accuracy score and GINI index on the PCA-reduced dataset for the random forest.

- **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.

- **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. As expected, on the PCA-reduced dataset, the boosted model considered all 20 PCs statistically significant. The variable importance table (Appendix C) show that PC1, PC2 and PC4 are again the top predictors.

- **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:

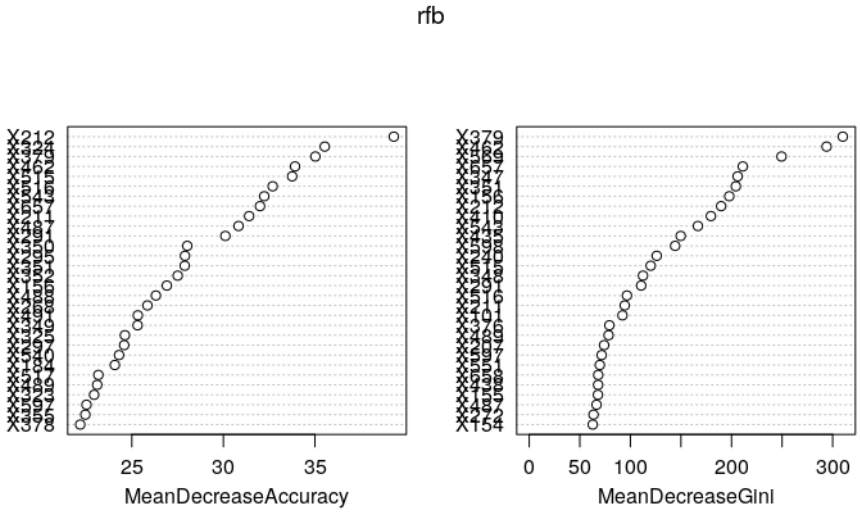

Figure 13: Variable importance plot by accuracy score and GINI index on the original dataset for the bagged trees.

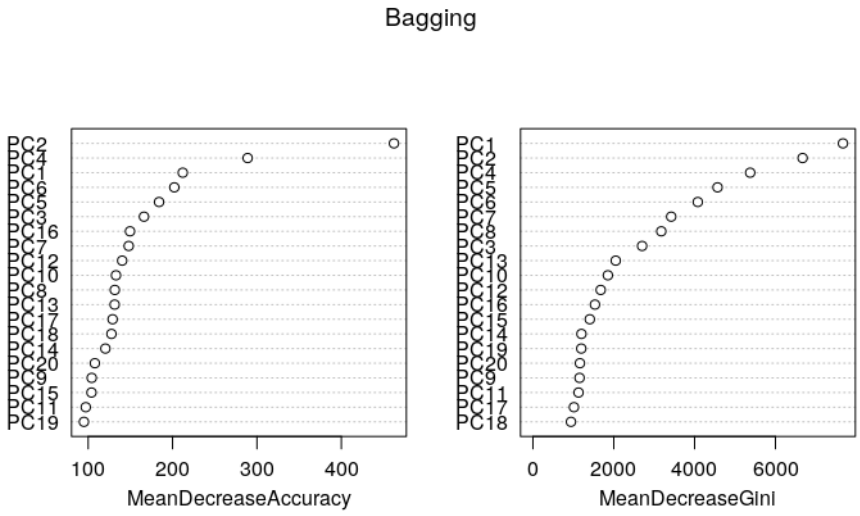

Figure 14: Variable importance plot by accuracy score and GINI index on the PCA-reduced dataset for the bagged trees.

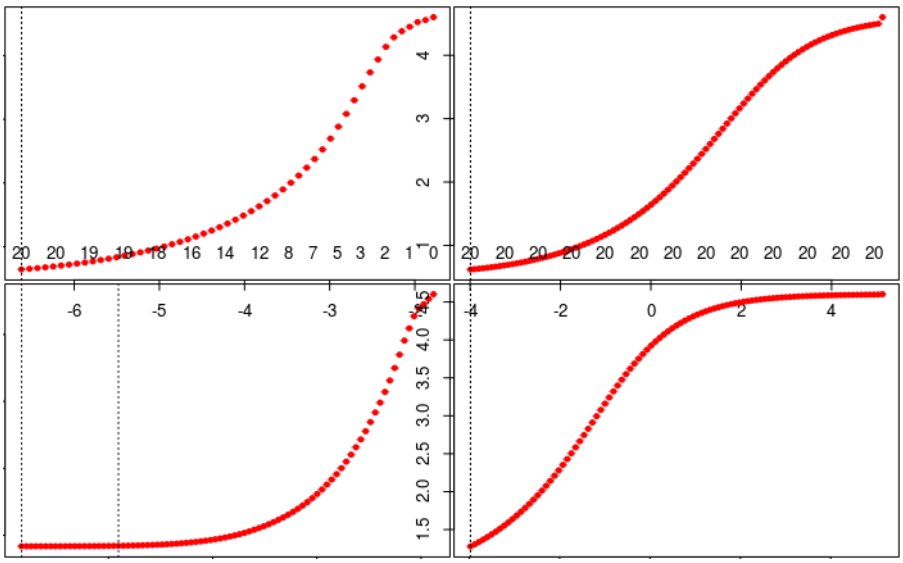

Figure 15: Ridge and Lasso regularization.

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(The regularization parameter) increases, the deviance(actually in our case, it functions same as misclassification rate) is monotonically increasing, which implies that 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.

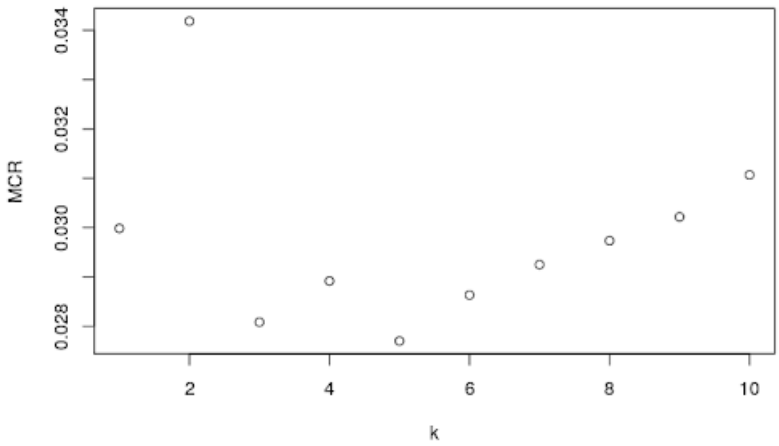

Figure 16: Choosing k value for KNN. The best k was chosen at 5.

- 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 and interaction terms in logistic regression. We chose the "radial" kernel (RBF) and achieved a **2.17%** misclassification 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, and we plot the decision boundary with only 2 predictors. We could plot this model with other predictors, which might generate different decision boundaries, but that'll be redundant to illustrate the model. If PC1 and PC2 are adequate to train a classifier with good predictive power and we train our model with PC1 and PC2. We think the plot above will have similar pattern as the PCA plot we get in early section.

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.

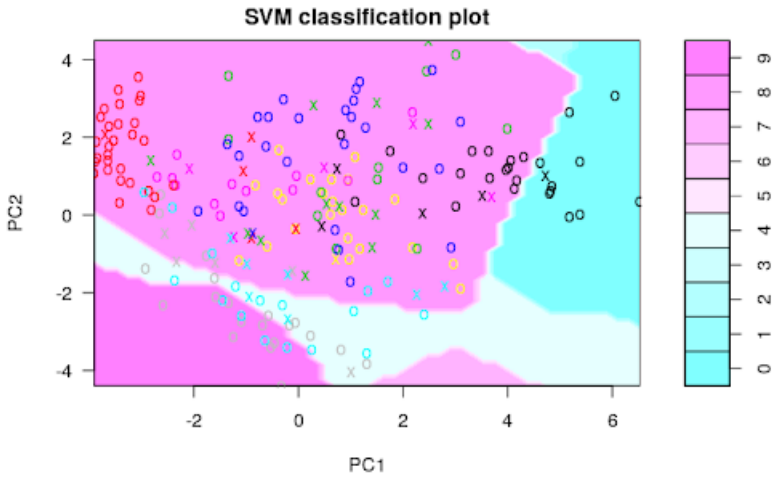

Figure 17: Visualization of SVM.

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

Review the results generated above and sythensize them in the context from which the data originated. What do the results tell your about your original research question? Are there any weaknesses that you see in your analysis? What additional questions would you explore next?

- 4-9 is the most difficult pair to predict across models
- No overfitting with PCA, but some with raw data
- Best model
- Compare with published models

References

At minimum, this will contain the full citation for your data set. If you reference existing analyses, they should be cited here as well.

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.