CMDA3654 Final Project

Brian Lee

Honor code:

"I have neither given nor received unauthorized assistance on this assignment." B.L.

I receive help from no one and give help to Cara Dunnavant.

Part I

Data description

The energy efficiency dataset contains data on the energy efficiency of 768 building shapes derived from 12 building shapes by simulation of various building characteristics such as orientation, glazing, and size. Energy efficiency is measured by the buildings' heating and cooling load requirements. This energy efficiency level is recorded with relevant data on each building, including two-dimensional and three-dimensional physical attributes.

This dataset is obtained from University of California Irvine's Machine Learning Repository. It was created by Angeliki Xifara and processed by Athanasios Tsanas.

```
## # A tibble: 6 x 10
##      X1    X2    X3    X4    X5    X6    X7    X8    Y1    Y2
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  0.98  514.  294   110.     7     2     0     0  15.6  21.3
## 2  0.98  514.  294   110.     7     3     0     0  15.6  21.3
## 3  0.98  514.  294   110.     7     4     0     0  15.6  21.3
## 4  0.98  514.  294   110.     7     5     0     0  15.6  21.3
## 5  0.9   564.  318.  122.     7     2     0     0  20.8  28.3
## 6  0.9   564.  318.  122.     7     3     0     0  21.5  25.4
```

In this situation, there are two response variables, the heating and cooling load on the buildings. There are eight explanatory variables. These explanatory variables include compactness of the building, building height, building surface area that of overall space, walls, and the roof. They also have the area and distribution of glazing on the building. Here is a preliminary look at the data frame.

Data visualization

In our exploratory data analysis, it is a good idea to visualize many variables of interest to assess the makeups of the data, both predictors and response, might have. This way, we gain insight to a variety of subjects of interest. By visualizing data, we can detect outliers on our predictors not for removal, but for awareness of which observations might be influencing the regression. We can also view the distributions of our data so we can generally see what makes a "typical" value for a predictor.

We can start by visualizing the value of every predictor against each predictor as well as response variables to scout out for multicollinearity or possible significant relationships between predictors and responses.

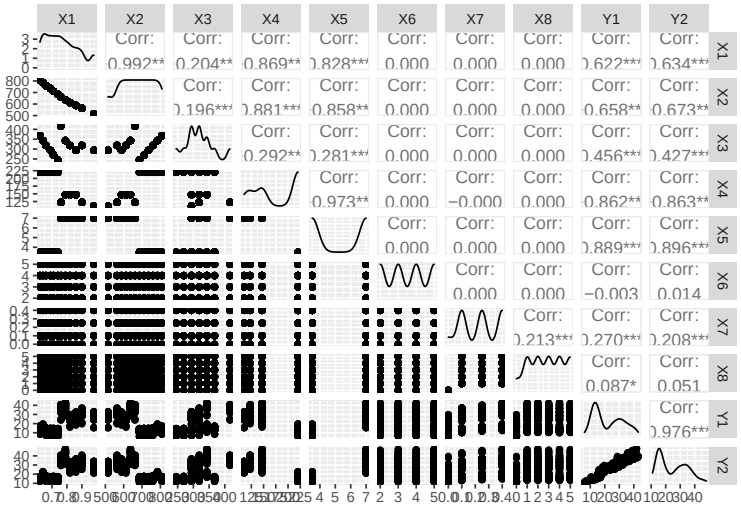

Figure 1: Matrix Plot of All Variables

Such a matrix plot can possibly be difficult to read. We can also visualize the distribution of variables of interest. We want to assess the general shape of the data to identify possible relationships. This will also help us detect outliers in our data.

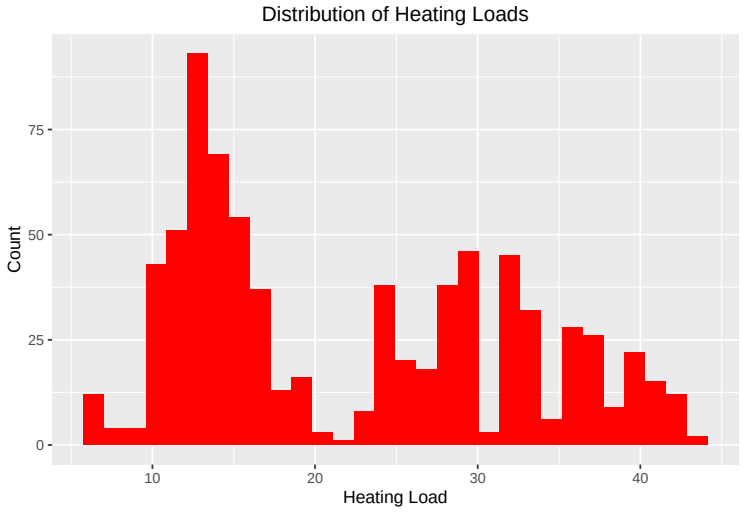

Figure 2: Various Predictors against other Predictors and Response Variables

We can comment on some insights gained from our data visualization.

- Examining both response variables, both the heating load and cooling load seem to favor lower values and seem far from evenly distributed.
- There is a higher mean response of heating load when building height, or glaze area increases, but there is a lower mean response of heating load when overall surface area increases. From this, we can reason that building size statistics does not necessarily correspond positively heating and cooling loads. In other words, one should not assume that a bigger building means higher heating or cooling

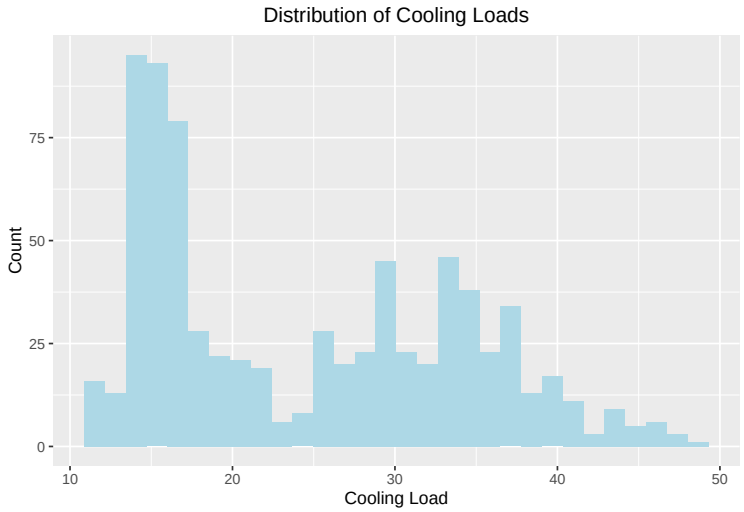

Figure 3: Various Predictors against other Predictors and Response Variables

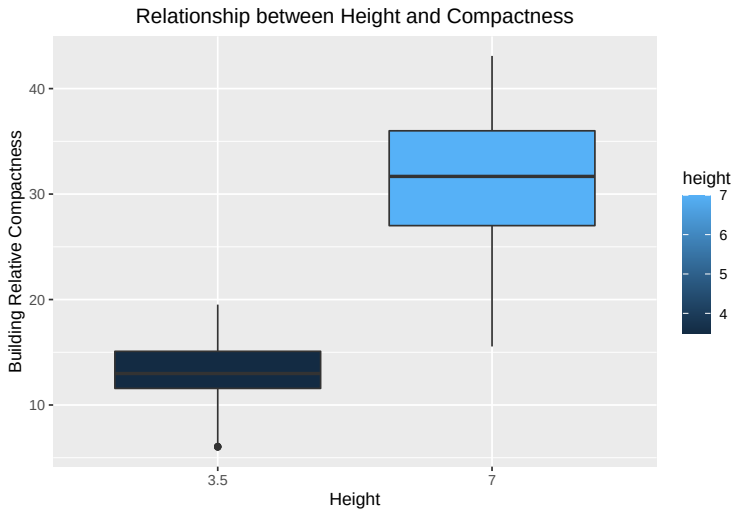

Figure 4: Various Predictors against other Predictors and Response Variables

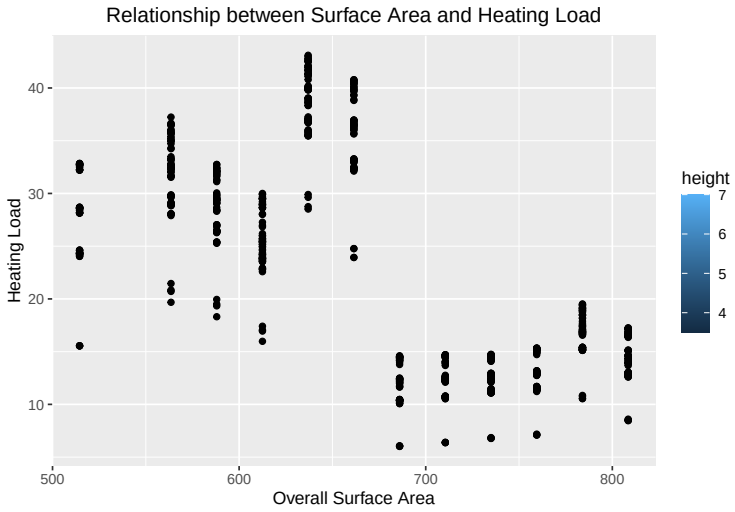

Figure 5: Various Predictors against other Predictors and Response Variables

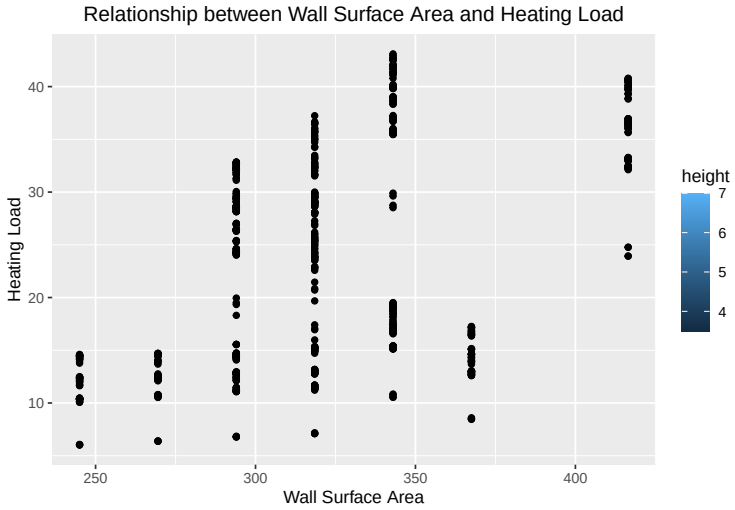

Figure 6: Various Predictors against other Predictors and Response Variables

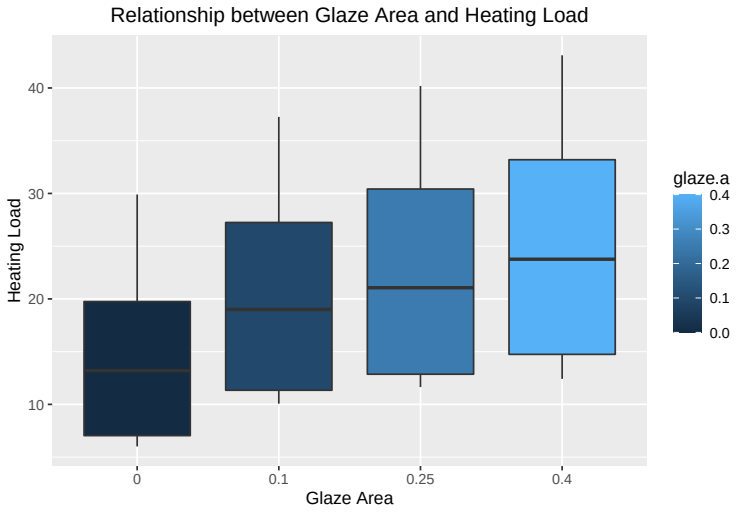

Figure 7: Various Predictors against other Predictors and Response Variables

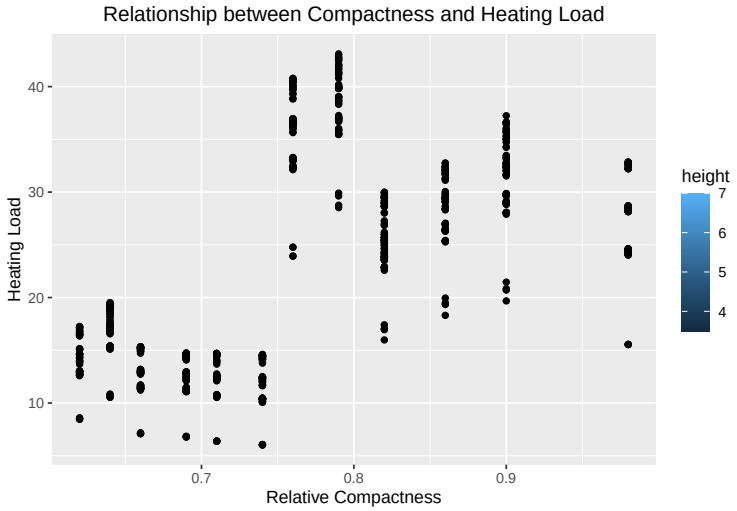

Figure 8: Various Predictors against other Predictors and Response Variables

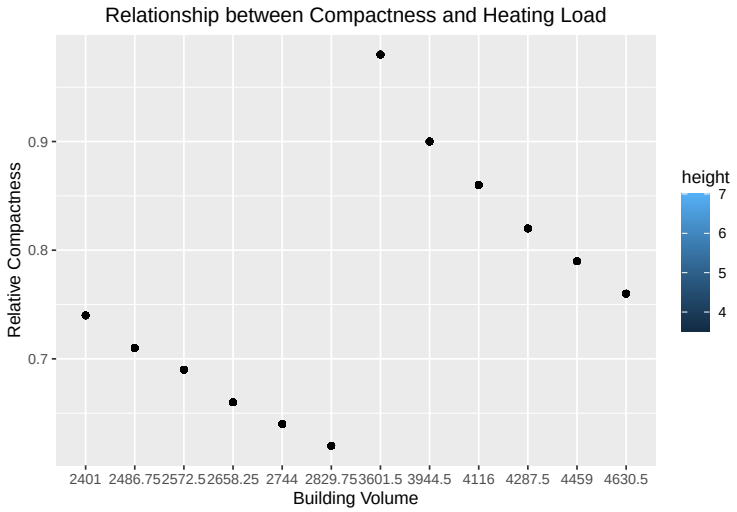

Figure 9: Various Predictors against other Predictors and Response Variables

load requirements. Perhaps heating and cooling load requirements, measures of energy efficiency, are dependent on physical efficiency as well.

- Compactness could possibly be an interesting variable to examine, as it takes into account both volume and overall size.

The goal, then is to construct models that can effectively predict the heating and cooling load scores required for each building. Given certain values for each of the building's potential explanatory variables, it should produce a value the resembles what a typical value for heating or cooling load should be. We can do this by constructing a multiple linear regression model. This is because area is a continuous response variable.

Because response variable cooling load and heating load both have similar relationships to the explanatory variables, only a multiple linear regression model on heating load will be constructed.

Model Setup

First, a linear regression model on heat load requirement is constructed by including all explanatory variables, X1 through X8.

```
##
## Call:
## lm(formula = Y1 ~ . - Y2, data = enb)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.8965 -1.3196 -0.0252  1.3532  7.7052
##
## Coefficients: (1 not defined because of singularities)
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  84.013418  19.033613   4.414 1.16e-05 ***
## X1          -64.773432  10.289448  -6.295 5.19e-10 ***
## X2           -0.087289   0.017075  -5.112 4.04e-07 ***
```

```
## X3            0.060813   0.006648   9.148  < 2e-16 ***
## X4                  NA         NA      NA       NA
## X5            4.169954   0.337990  12.338  < 2e-16 ***
## X6           -0.023330   0.094705  -0.246  0.80548
## X7           19.932736   0.813986  24.488  < 2e-16 ***
## X8            0.203777   0.069918   2.915  0.00367 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.934 on 760 degrees of freedom
## Multiple R-squared:  0.9162, Adjusted R-squared:  0.9154
## F-statistic:  1187 on 7 and 760 DF,  p-value: < 2.2e-16
```

Next, we utilize Akaike's Information Criterion method (AIC), which will iterate until it a minimum AIC score is reached. In this way an optimal set of explanatory variables is obtained. These are very often statistically significant as well. As can be seen, the optimal set of predictors to predict heat load is relative compactness, surface area, wall area, height, glazing area, and distribution of glazing.

We can see if we can tune our model to be even more optimal with the addition of interaction terms or higher order terms as predictors. This can be done through trial and error.

Result

```
##
## Call:
## lm(formula = Y1 ~ X1 + X2 + X3 + X5 + X7 + X8 + X3:X2 + X7:X8 +
##     X1:X2 + X1:X5, data = enb)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.2156 -1.2059  0.3352  1.6056  5.1556
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.913e+03  1.204e+02 -15.888  < 2e-16 ***
## X1           1.226e+03  8.933e+01  13.726  < 2e-16 ***
## X2           1.688e+00  1.179e-01  14.314  < 2e-16 ***
## X3           1.045e+00  7.498e-02  13.938  < 2e-16 ***
## X5           6.680e+01  4.602e+00  14.514  < 2e-16 ***
## X7           2.706e+01  1.192e+00  22.695  < 2e-16 ***
## X8           7.883e-01  9.927e-02   7.940 7.28e-15 ***
## X2:X3       -1.672e-03  1.236e-04 -13.521  < 2e-16 ***
## X7:X8       -2.860e+00  3.944e-01  -7.251 1.02e-12 ***
## X1:X2       -3.410e-01  8.272e-02  -4.122 4.17e-05 ***
## X1:X5       -7.072e+01  5.425e+00 -13.037  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.432 on 757 degrees of freedom
## Multiple R-squared:  0.9427, Adjusted R-squared:  0.9419
## F-statistic:  1245 on 10 and 757 DF,  p-value: < 2.2e-16
```

As a result, our regression equation is given by the following:

$$\hat{y} = -5.622 + 9.553X1 + 0.3756X2 + 0.4836X3 + 7.041X5 + 2.706X7 +$$

$$0.7883X8 - 0.0006714X2X3 - 2.86X7X8 + 0.4082X1X2 - 7.072X1X5 \quad (1)$$

It should be explained as to why certain terms were included or not included after running the Information Criteria. As for the interaction terms, these were included on the basis of how they would interact with each other in a physical interpretation. For example, perhaps a better model might be found by including an interaction to represent the real-life interaction between the compactness of a building and its surface area.

- An interaction between surface area and wall area was included because it makes sense that the overall surface area changes when one surface changes in its area.
- An interaction term between glazing area and glazing area distribution was included because how glaze is distributed must be accounted for, not just its total area.
- An interaction term between compactness and surface area is included because a more compact building would have less surface area.
- An interaction term between compactness and height is included because a building that is very tall is less compact.

It was decided not to include any higher-order and nonlinear terms in the data because upon viewing the matrix plot where each predictor is plotted against the response variable, heating load, it is not at all clear if any one predictor acts in a nonlinear fashion against the response variable.

This model is quite satisfactory because not only is every predictor's coefficient statistically significant, we have an even higher R^2 and R^2adj value of 94.27% and 94.19%, respectively. In other words, around 94.27% of the variability in the data is explained by the model created.

Part II

Data description

The MNIST dataset is a database of 60,000 images of handwritten digits from 0 to 9, derived from the NIST Special Database. To clarify, each image exists with a size of 28x28 pixels, and each pixel takes on a value between 0.0 to 1.0 as the value on the grayscale.

The MNIST database of handwritten digits was constructed by Yann LeCun, Corinna Cortes, and Christopher J.C. Burges.

The high-dimensionality of the MNIST dataset can be visualized by constructing a t-SNE plot that gives an intuitive idea as to how the data is arranged. t-SNE reduces this high dimensionality to produce this plot.

```
## Performing PCA
## Read the 60000 x 50 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
##  - point 10000 of 60000
##  - point 20000 of 60000
##  - point 30000 of 60000
##  - point 40000 of 60000
##  - point 50000 of 60000
```

```
##  - point 60000 of 60000
## Done in 501.27 seconds (sparsity = 0.002086)!
## Learning embedding...
## Iteration 50: error is 118.896811 (50 iterations in 18.55 seconds)
## Iteration 100: error is 118.896811 (50 iterations in 19.85 seconds)
## Iteration 150: error is 118.833437 (50 iterations in 28.87 seconds)
## Iteration 200: error is 108.762441 (50 iterations in 24.39 seconds)
## Fitting performed in 91.67 seconds.
```

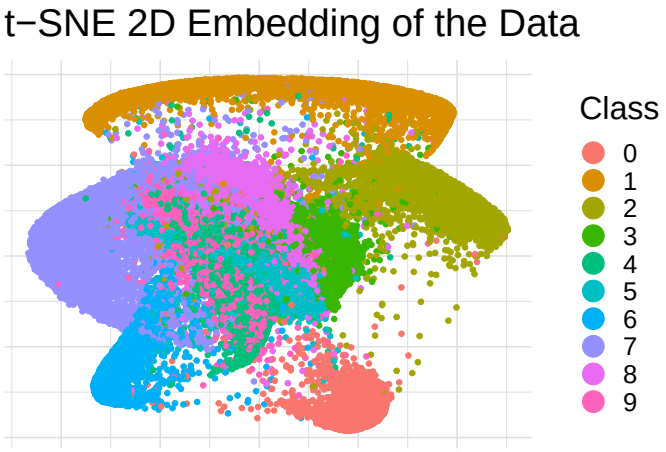

Figure 10: t-SNE Plot of Data Embedding, 2D

In this dataset, the response variable would be a categorical variable of digits 0 through 9. In other words, these would be what the guesses and conclusions our models arrive at on which digit is being resembled. The predictors are the grayscale values of each pixel in the 784 pixel image. This is because the values of the pixels determine how the model categorizes the input image.

The goal with this dataset, then, is to build a model that can accurately classify what digit has been handwritten in the image. With the understanding that individuals have variability in their handwriting, it would be helpful to create a model that can accommodate those differences when determining what digit is handwritten in the image. If we are able to construct such a model that can classify which digit is which, we have tremendous utility in real-life settings that may require a checking of handwriting, such as auto-grading for academia or text parsing.

Dimension Reduction and LDA Setup

Because the problem presented by the MNIST dataset is one of classification, one classification method that can be constructed is through dimension reduction and subsequent linear discrimination analysis. First, the training and testing data for the predictor must be reshaped into a matrix and scaled by 255 in order that the values in said matrix are between 0 and 1.

After reshaping and rescaling the data, we perform principle component analysis to reduce the dimensional space. This is done using the training dataset. In addition, through making a scree plot, the number of principle components is determined.

Seeing as that there are more than a binomial outcome space for our classification, linear discriminant analysis, or LDA, should be used as a classifier after the dimension reduction. And looking at the screeplot,

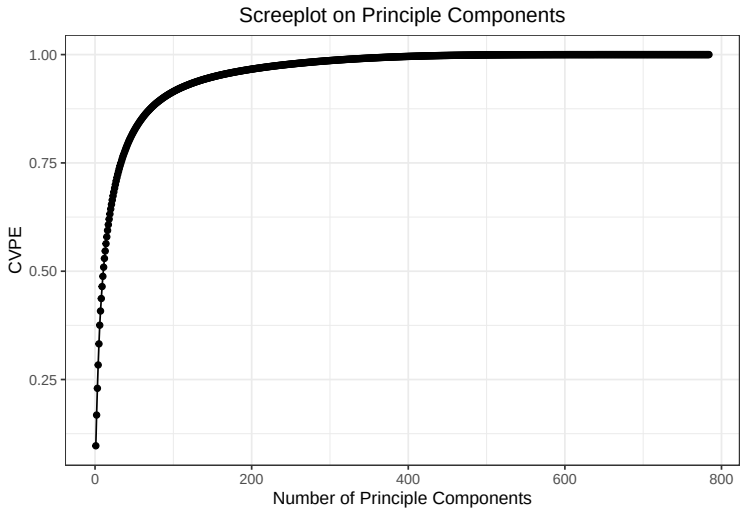

Figure 11: PCA Screeplot

the leveling of cumulative proportion of variance explained occurs around 150 to 175 principle components, so the number of principle compenents to use should exist somewhere within that range.

New predictors must be obtained in order to construct a new classifier. The scores from a table of principle component analysis scores after performing the dimension reduction will be used. They are stored in a new training dataset and serve as the new predictors for the LDA classifier, which is then fit to said training data.

After constructing a classification method through applying LDA to the training dataset of PCA scores, we run it against the testing data to make predictions and also see how satisfactory the classifer is.

Dimension Reduction and QDA Result

A receiver operating characteristic curve, or ROC curve, can assess the effectiveness of the classifier. It is the measure of the true positive versus the false positive rate. In this case, the ROC curve is constructed on the linear discriminant classifier.

As can be seen, the area under the ROC curve is 0.9997, suggesting that the linear discriminant analysis applied to the dimension reduction is very effective.

Shrinkage Regression Setup

For the MNIST dataset, it would also make sense to run a shrinkage regression. This shrinkage regression can perform

In this case, the LASSO shrinkage regression will be used because it can perform variable selection and regularization to arrive at enhanced predictions. The shrinkage regressions are then plotted.

We are also to choose a tuning parameter through cross-validation. Selecting a good tuning parameter is essential to fit a model that can classify the data.

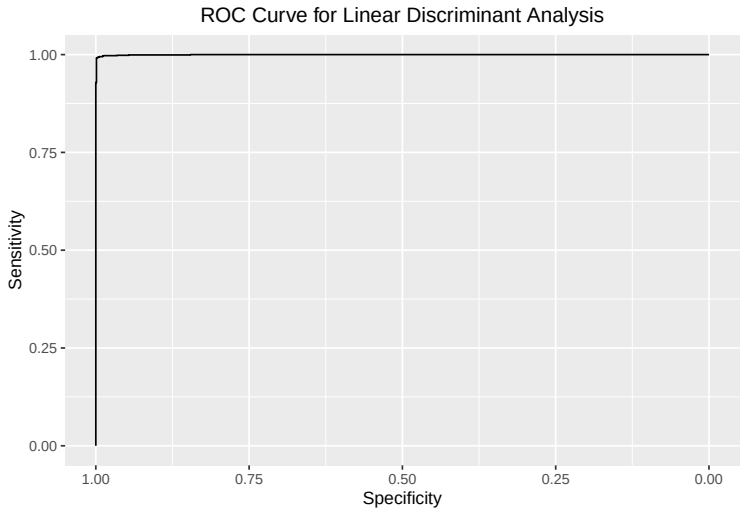

Figure 12: ROC Curve on LDA

Shrinkage Regression Result

A receiver operating characteristic curve, or ROC curve, can assess the effectiveness of the classifier. It is the measure of the true positive versus the false positive rate. In this case, the ROC curve is constructed on the LASSO shrinkage regression model we constructed.

As can be seen, the area under the ROC curve is 0.8850, suggesting that the LASSO shrinkage regression applied to the data is quite effective.

Convolutional Neural Network Setup

The MNIST dataset contains training and testing datasets for the predictor and categorical response variables. In the case of images, it makes the most sense to use convolutional neural network which view the 28x28 pixel as a convolutional layer on which we apply filters to. Therefore, the training and testing data must be reshaped to be 28x28 layers.

For the MNIST dataset, it would also make sense to construct a neural network. The neural network should be able to take in input neurons and iterate through various hidden layers to arrive at a conclusion of what digit is presented by the MNIST data. Here, each neuron in the input layer corresponds to a pixel in the 28x28 image. This means that we would have 784 neurons in the input layer, structured in a square layer of 28x28 neurons.

```
## Model: "sequential"
## _________________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =========================================================================
## conv2d_1 (Conv2D)            (None, 27, 27, 32)        160
## _________________________________________________________________________
## max_pooling2d_1 (MaxPooling2D)  (None, 13, 13, 32)     0
## _________________________________________________________________________
## conv2d (Conv2D)              (None, 12, 12, 64)        8256
## _________________________________________________________________________
## dropout_2 (Dropout)          (None, 12, 12, 64)        0
## _________________________________________________________________________
```

```
## max_pooling2d (MaxPooling2D)        (None, 6, 6, 64)              0
## _________________________________________________________________________
## dropout_1 (Dropout)                 (None, 6, 6, 64)              0
## _________________________________________________________________________
## flatten (Flatten)                   (None, 2304)                  0
## _________________________________________________________________________
## dense_1 (Dense)                     (None, 98)                    225890
## _________________________________________________________________________
## dropout (Dropout)                   (None, 98)                    0
## _________________________________________________________________________
## dense (Dense)                       (None, 10)                    990
## =========================================================================
## Total params: 235,296
## Trainable params: 235,296
## Non-trainable params: 0
## _________________________________________________________________________
```

The neural network is then compiled and trained using the training datasets on predictor and response variables in order to assign fair weights and bias values to each hidden layer.

Neural Network Result

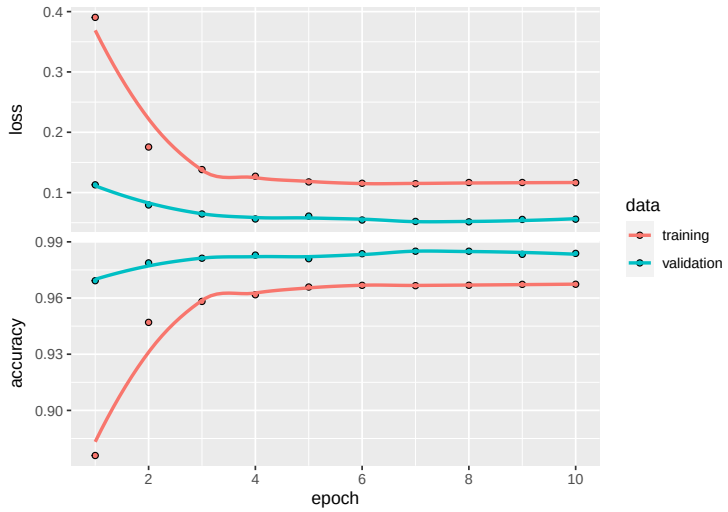

Figure 13: Accuracy and Loss of Convolutional Neural Network

```
## Test loss: 0.04952027
```

```
## Test accuracy: 0.9845
```

We can see that the accuracy is around 98%, and the loss is around 4%. Although this is most likely a solid model that can adequately classify the images in the dataset, this raises questions of over-fitting. Such questions can be explored by fine-tuning the network.

Code

```
#Read dataset
library(readxl)
enb = read_excel("./ENB2012_data.xlsx")
head(enb)

compact = enb$X1
surf.a = enb$X2
wall.a = enb$X3
roof.a = enb$X4
height = enb$X5
orient = enb$X6
glaze.a = enb$X7
glaze.a.dist = enb$X8

load.heat = enb$Y1
load.cool = enb$Y2

library(GGally)
ggpairs(enb)

library(ggplot2)

ggplot(enb) + geom_histogram(fill="red",aes(load.heat))
+ ggtitle("Distribution of Heating Loads")
+ xlab("Heating Load") + ylab("Count")
+ theme(plot.title = element_text(hjust=0.5))

ggplot(enb) + geom_histogram(fill="lightblue", aes(load.cool))
+ ggtitle("Distribution of Cooling Loads")
+ xlab("Cooling Load") + ylab("Count")
+ theme(plot.title = element_text(hjust=0.5))

ggplot(enb, aes(x=as.factor(height), y=load.heat, fill=height)) + geom_boxplot()
+ ggtitle("Relationship between Height and Compactness")
+ xlab("Height") + ylab("Building Relative Compactness")
+ theme(plot.title = element_text(hjust=0.5))

ggplot(enb, aes(x=surf.a, y=load.heat, fill=height))
+ geom_point() + ggtitle("Relationship between Surface Area and Heating Load")
+ xlab("Overall Surface Area") + ylab("Heating Load")
+ theme(plot.title = element_text(hjust=0.5))

ggplot(enb, aes(x=wall.a, y=load.heat, fill=height)) + geom_point()
+ ggtitle("Relationship between Wall Surface Area and Heating Load")
+ xlab("Wall Surface Area") + ylab("Heating Load")
+ theme(plot.title = element_text(hjust=0.5))


  ggplot(enb, aes(x=as.factor(glaze.a), y=load.heat, fill=glaze.a)) + geom_boxplot()
  + ggtitle("Relationship between Glaze Area and Heating Load")
  + xlab("Glaze Area") + ylab("Heating Load")
```

```
  + theme(plot.title = element_text(hjust=0.5))

  ##Perhaps more interesting?
  ggplot(enb, aes(x=compact, y=load.heat, fill=height)) + geom_point()
  + ggtitle("Relationship between Compactness and Heating Load")
  + xlab("Relative Compactness") + ylab("Heating Load")
  + theme(plot.title = element_text(hjust=0.5))

  ggplot(enb, aes(x=as.factor(height*surf.a), y=compact, fill=height)) + geom_point()
  + ggtitle("Relationship between Compactness and Heating Load")
  + xlab("Building Volume") + ylab("Relative Compactness")
  + theme(plot.title = element_text(hjust=0.5))

model.mlr.all = lm(data=enb, Y1 ~ .-Y2)
summary(model.mlr.all)

#AIC
model.mlr.aicmin = step(model.mlr.all, direction="backward")
summary(model.mlr.aicmin)

model.optimal = lm(formula = Y1 ~ X1 + X2 + X3 + X5
+ X7 + X8 + X3:X2 + X7:X8 + X1:X2 + X1:X5, data = enb)
summary(model.optimal)

##########################PART2#########################
#Load packages
library(keras)
library(Rtsne)
library(ggplot2)
library(MASS)
library(glmnet)
library(pROC)


mnist = dataset_mnist()

x.train = mnist$train$x
x.test = mnist$test$x
y.train = mnist$train$y
y.test = mnist$test$y
y.train = as.factor(y.train)
y.test = as.factor(y.test)

x.train = array_reshape(x.train, c(nrow(x.train), 784))
x.test = array_reshape(x.test, c(nrow(x.test), 784))
x.train = x.train/255
x.test = x.test/255

x.train.tsne = x.train
y.train.tsne = y.train
x.test.tsne = x.test
y.test.tsne = y.test
```

```
#Reshape and rescale data
x.train.tsne = array_reshape(x.train.tsne, c(nrow(x.train.tsne), 784))
x.test.tsne = array_reshape(x.test.tsne, c(nrow(x.test.tsne), 784))
x.train.tsne =  x.train.tsne[,apply(x.train.tsne, 2, var, na.rm=TRUE) != 0]
x.train.tsne = x.train.tsne/255
x.test.pca = x.test.tsne/255

mnist.tsne = Rtsne(x.train.tsne, dims=2, perplexity=30, verbose=TRUE, max_iter = 200)
embedding = as.data.frame(mnist.tsne$Y)
embedding$Class = as.factor(y.train.tsne)

ggplot(embedding, aes(x=V1, y=V2, color=Class)) +
   geom_point(size=1.25) +
   guides(colour = guide_legend(override.aes = list(size=6))) +
   xlab("") + ylab("") +
   ggtitle("t-SNE 2D Embedding of the Data") +
   theme_light(base_size=20) +
   theme(strip.background = element_blank(),
         strip.text.x     = element_blank(),
         axis.text.x      = element_blank(),
         axis.text.y      = element_blank(),
         axis.ticks       = element_blank(),
         axis.line        = element_blank(),
         panel.border     = element_blank())

#Separate into training and test sets
x.train.pca = x.train
x.test.pca = x.test
y.train.pca = y.train
y.test.pca = y.train

#Reshape and rescale the data for dimension reduction
x.train.pca = array_reshape(x.train.pca, c(nrow(x.train.pca), 784))
x.test.pca = array_reshape(x.test.pca, c(nrow(x.test.pca), 784))
x.train.pca = x.train.pca/255
x.test.pca = x.test.pca/255

#Perform PCA to reduce the dimension
mnist.pca = princomp(x.train.pca)

#Know variance for screeplot
var = sum(diag(cov(x.train.pca)))
df = data.frame(npc = 1:ncol(x.train.pca), cpve = cumsum(mnist.pca$sdev^2)/var)

ggplot(df, aes(npc, cpve)) + geom_line() + geom_point() + theme_bw()
+ ggtitle("Screeplot on Principle Components") + ylab("CVPE")
+ xlab("Number of Principle Components")
+ theme(plot.title = element_text(hjust=0.5))

num_pc = 150;

#Store PCA scores
mnist.pca.scores = as.data.frame(mnist.pca$scores)
```

```r
#Train with new training set to make classifier based on PC score, LDA
train.lda = as.data.frame(mnist.pca.scores[,1:num_pc])
train.lda$Y = y.train.pca

lda.fit = lda(Y ~ ., data = train.lda)

#Create test the data to assess fit and test it using
test.lda = as.data.frame(x.test.pca %*% mnist.pca$loadings[,1:num_pc])
lda.pred = predict(lda.fit, test.lda)

#Represent assessment using ROC
lda.roc = roc(y.test, lda.pred$posterior[,2])
lda.roc

ggroc(lda.roc) + ggtitle("ROC Curve for Linear Discriminant Analysis")
+ xlab("Specificity") + ylab("Sensitivity")
+ theme(plot.title = element_text(hjust=0.5))

mnist = dataset_mnist()

#Extract MNIST's training and test sets.
x.train.lasso = mnist$train$x
y.train.lasso = as.factor(mnist$train$y)
x.test.lasso = mnist$test$x
y.test.lasso = as.factor(mnist$test$y)

#Reshape and rescale the data for dimension reduction
x.train.lasso = array_reshape(x.train.lasso, c(nrow(x.train.lasso), 784))
x.test.lasso = array_reshape(x.test.lasso, c(nrow(x.test.lasso), 784))
x.train.lasso = x.train.lasso/255
x.test.lasso = x.test.lasso/255

lasso.fit = glmnet(x.train.lasso, y.train.lasso, family="multinomial", alpha=1)
plot(lasso.fit, xvar="lambda")

#LASSO FREEZES

cv.lasso = cv.glmnet(x.train.lasso, y.train.lasso, type.measure="mse",
  family="multinomial", alpha=1)
plot(cv.lasso)

#LASSO FREEZES

lasso.pred = predict(lasso.fit, s=cv.lasso$lambda.1se,
  newx=x.test.lasso, type="response")

Represent assessment using ROC
lasso.roc = roc(y.test.lasso, lasso.pred, levels=c("No", "Yes"))
lasso.roc

ggroc(lasso.roc)

#LASSO FREEZES
```

```
mnist = dataset_mnist()
img_rows <- 28
img_cols <- 28

#Extract MNIST's training and test sets.
x.train = mnist$train$x
y.train = mnist$train$y

x.test = mnist$test$x
y.test = mnist$test$y

x.train <- array_reshape(x.train, c(nrow(x.train), img_rows, img_cols, 1))
x.test <- array_reshape(x.test, c(nrow(x.test), img_rows, img_cols, 1))
input_shape <- c(img_rows, img_cols, 1)

#Normalize the x variables to be on a 0,1 scale.
x.train = x.train/255
x.test = x.test/255

y.train.pca = y.train
#Make y variables categorical
y.train = to_categorical(y.train, 10)
y.test = to_categorical(y.test, 10)

model.cnn = keras_model_sequential()
model.cnn %>%
  layer_conv_2d(filters=32,kernel_size = c(2,2), input_shape = c(28,28,1),
    activation="relu") %>%
  layer_max_pooling_2d(pool_size=c(2,2), strides=c(2,2)) %>%

  layer_conv_2d(filters=64, kernel_size = c(2,2), activation="relu") %>%
  layer_dropout(rate=0.45) %>%

  layer_max_pooling_2d(pool_size = c(2,2), strides=c(2,2)) %>%
  layer_dropout(rate=0.45)%>%

  layer_flatten() %>%
  layer_dense(98, activation="relu") %>%
  layer_dropout(rate=0.45) %>%
  layer_dense(10, activation="softmax")

summary(model.cnn)

model.cnn %>% compile(loss="categorical_crossentropy",
                optimizer=optimizer_rmsprop(),
                metrics=c("accuracy"))

history.rms = model.cnn %>% fit(x=x.train, y=y.train, epochs=20,
  batch_size=64, validation_split=0.2)

  plot(history.rms)
```

```
scores <- model.cnn %>% evaluate(
x.test, y.test, verbose = 0
)

# Output metrics
cat('Test loss:', scores[[1]], '\n')
cat('Test accuracy:', scores[[2]], '\n')
```

References

- Energy efficiency dataset: https://archive.ics.uci.edu/ml/datasets/Energy+efficiency
- MNIST database documentation: http://yann.lecun.com/exdb/mnist/
- Video on neural networks - 3Blue1Brown: https://www.youtube.com/watch?v=aircAruvnKk
- t-SNE algorithm explanation: https://towardsdatascience.com/an-introduction-to-t-sne-with-python-example-5a3a293108d1
- PCA explanation: https://builtin.com/data-science/step-step-explanation-principal-component-analysis
- MLR variable transformation resource: https://rpubs.com/mpfoley73/495822