```r
# This R environment comes with many helpful analytics packages
installed
# It is defined by the kaggle/rstats Docker image:
https://github.com/kaggle/docker-rstats
# For example, here's a helpful package to load

library(tidyverse) # metapackage of all tidyverse packages

# Input data files are available in the read-only "../input/"
directory
# For example, running this (by clicking run or pressing Shift+Enter)
will list all files under the input directory

list.files(path = "../input")

# You can write up to 20GB to the current directory (/kaggle/working/)
that gets preserved as output when you create a version using "Save &
Run All"
# You can also write temporary files to /kaggle/temp/, but they won't
be saved outside of the current session
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.1.2     ✔ readr     2.1.4
✔ forcats   1.0.0     ✔ stringr   1.5.0
✔ ggplot2   3.4.2     ✔ tibble    3.2.1
✔ lubridate 1.9.2     ✔ tidyr     1.3.0
✔ purrr     1.0.1
── Conflicts ────────────────────────────────────────── tidyverse conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ⓘ Use the conflicted package (<http://conflicted.r-lib.org/>) to
force all conflicts to become errors

[1] "digit-recognizer"
```

Purpose: This script is written to complete a Kaggle Machine Learning competition called Digit Recognizer. The intent is to perform accurate machine learning for MNIST (Modified National Institute of Standards and Technology). The domain for this dataset is computer vision. The data is unstructed as hand-written images of digits. This model will learn to accurately detect digits from hand-written images. This analysis is originally structured in R but can also be executed in Python. The training set contains 60,000 images while the test set has 10,000 images.

Why/ Background: (https://www.nist.gov/about-nist)MNIST data serves as a foundational resource for the US to improve international competitiveness for innovation and improving the economy. Established in 1901, its intent is to improve the US's competitiveness, as at some point the nation was considered second-rate to other countries for innovation. Strengthening innovation impacts the nation's economic security and the quality of life of its citizens. This specific data is used for classification algorithms.

(https://learn.microsoft.com/en-us/azure/open-datasets/dataset-mnist?tabs=azureml-opendatasets NEXT) The data has been formatted so that all images are size-normalized and centered for recognition. Each image is 28x28 (784 pixels), with pixel values ranging from 0 to 255. Pixels are usually formatted as byte images. In these images, 0 is seen as black while 255 is white. This allows color intensity to be distinguishable.

Load data

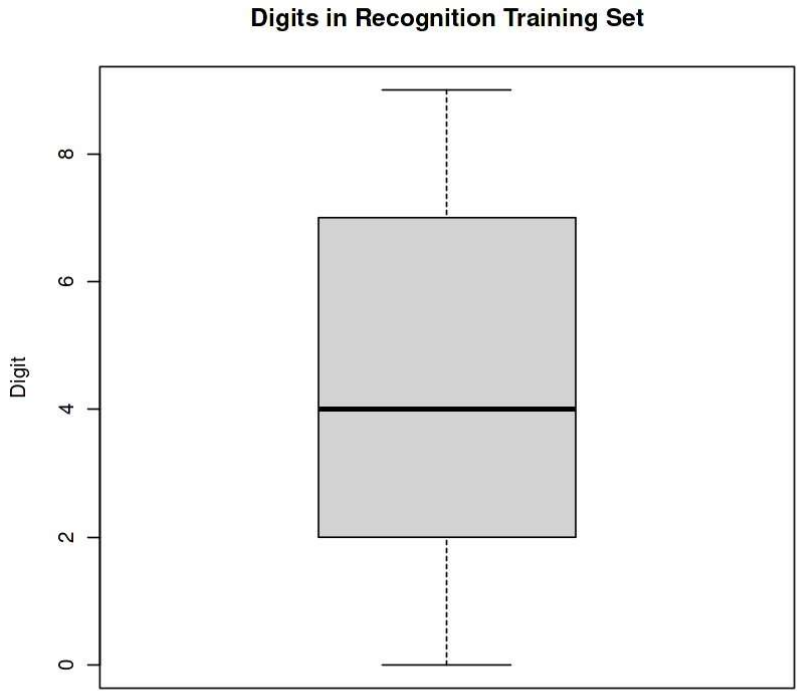
```
library(readr)
test_data <- read_csv("/kaggle/input/digit-recognizer/test.csv")
train_data <- read_csv("/kaggle/input/digit-recognizer/train.csv")
#head(summary(train_data))
#class(train_data)
str(test_data)
(train_data$label)
```

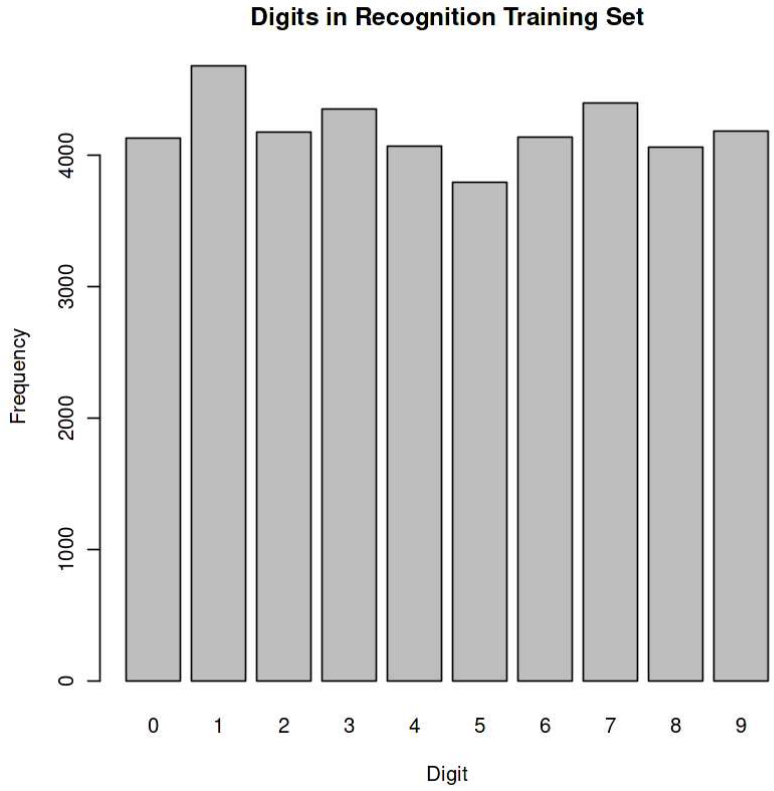
```
Rows: 28000 Columns: 784
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
dbl (784): pixel0, pixel1, pixel2, pixel3, pixel4, pixel5, pixel6,
pixel7, p...

ⓘ Use `spec()` to retrieve the full column specification for this
data.
ⓘ Specify the column types or set `show_col_types = FALSE` to quiet
this message.
Rows: 42000 Columns: 785
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
dbl (785): label, pixel0, pixel1, pixel2, pixel3, pixel4, pixel5,
pixel6, pi...

ⓘ Use `spec()` to retrieve the full column specification for this
data.
ⓘ Specify the column types or set `show_col_types = FALSE` to quiet
this message.

spc_tbl_ [28,000 × 784] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ pixel0  : num [1:28000] 0 0 0 0 0 0 0 0 0 0 ...
 $ pixel1  : num [1:28000] 0 0 0 0 0 0 0 0 0 0 ...
 $ pixel2  : num [1:28000] 0 0 0 0 0 0 0 0 0 0 ...
 $ pixel3  : num [1:28000] 0 0 0 0 0 0 0 0 0 0 ...
 $ pixel4  : num [1:28000] 0 0 0 0 0 0 0 0 0 0 ...
 $ pixel5  : num [1:28000] 0 0 0 0 0 0 0 0 0 0 ...
 $ pixel6  : num [1:28000] 0 0 0 0 0 0 0 0 0 0 ...
 $ pixel7  : num [1:28000] 0 0 0 0 0 0 0 0 0 0 ...
 $ pixel8  : num [1:28000] 0 0 0 0 0 0 0 0 0 0 ...
 $ pixel9  : num [1:28000] 0 0 0 0 0 0 0 0 0 0 ...
```

```
[41617] 7 8 1 6 7 8 8 1 6 2 0 9 1 2 7 7 7 7 5 6 0 0 4 0 9 0 9 5 5 7 7
7 6 7 3 7
[41653] 9 4 3 1 9 6 4 1 3 1 6 9 8 0 3 9 1 1 3 6 8 9 3 7 3 5 2 5 8 7 0
0 5 1 8 8
[41689] 5 8 6 4 2 2 0 9 1 9 6 9 8 0 2 4 4 1 8 0 6 6 0 3 8 6 6 8 5 3 4
9 2 6 0 3
[41725] 6 0 4 3 7 1 3 6 4 4 7 7 4 3 9 1 7 4 3 6 9 3 5 0 1 9 3 5 8 9 1
5 2 1 5 1
[41761] 3 9 1 3 6 6 3 3 2 3 9 0 2 0 0 9 3 1 6 8 3 3 6 7 3 3 0 8 9 2 0
5 2 3 6 9
[41797] 1 4 4 6 0 4 7 3 3 6 5 2 4 4 8 9 1 1 7 9 2 8 6 2 1 7 9 4 7 8 1
0 6 3 2 1
[41833] 9 7 0 6 5 0 4 6 8 8 2 4 4 7 3 2 6 9 7 1 1 0 1 2 7 4 3 9 8 1 1
9 3 5 1 7
[41869] 3 6 2 4 4 6 4 3 0 8 1 4 1 3 1 7 3 2 0 3 8 1 2 6 7 2 3 3 1 4 0
1 4 9 2 8
[41905] 1 5 5 5 7 8 6 1 5 6 9 5 8 2 3 8 8 6 7 1 4 5 1 6 5 8 0 2 1 1 7
3 1 2 1 1
[41941] 4 5 5 4 9 7 9 4 8 7 4 1 1 4 1 3 4 7 2 9 1 8 9 2 8 2 4 6 0 9 2
3 4 4 3 9
[41977] 2 4 4 4 7 2 8 7 3 3 0 5 0 5 3 1 9 6 4 0 1 7 6 9
```

The datasets provided are dataframes. The test data has 28,000 rows and 784 columns. The training data has 42,000 rows and 785 columns. The additional column in the training dataset is an attribute named "label". This column individually lists digits 0 to 9.

```
boxplot(train_data$label, main= "Digits in Recognition Training Set",
ylab= "Digit")
train_labels <- table(train_data$label)
barplot(train_labels, main= "Digits in Recognition Training Set",
        ylab= "Frequency", xlab= "Digit")
```


Digits in Recognition Training Set
Digit
0
2
4
6
8

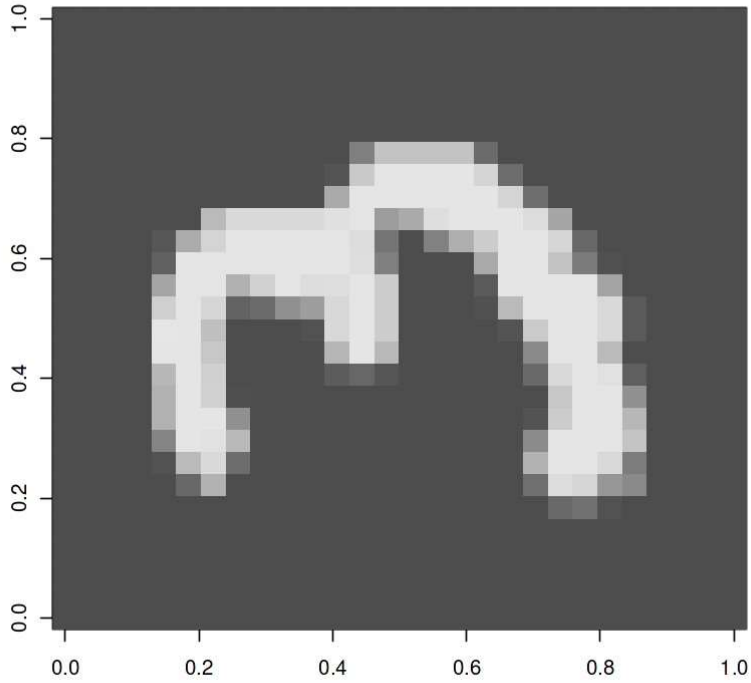

Digits in Recognition Training Set
Frequency
0
1000
2000
3000
4000
0
1
2
3
4
5
6
7
8
9
Digit

Let's get a general look at the hand-written images. The images are stored as matrices of pixelated values. Subset the training data to make a 28x28 matrix with the labels removed.

```
#https://www.kaggle.com/code/kobakhit/digital-recognizer-in-r
matrix1 = matrix(unlist(train_data[10,-1]),nrow = 28,byrow = T)
matrix1
# Plot that matrix
image(matrix1,col=grey.colors(255))
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
[,13]
 [1,] 0    0    0    0    0    0    0    0    0    0    …        0
0
 [2,] 0    0    0    0    0    0    0    0    0    0    …        0
0
 [3,] 0    0    0    0    0    0    0    0    0    0    …        0
0
 [4,] 0    0    0    0    0    0    0    0    0    0    …        0
0
 [5,] 0    0    0    0    0    0    0    5   60  136    …        0
0
 [6,] 0    0    0    0    0    0   25  152  253  253    …        0
0
 [7,] 0    0    0    0    0    0  135  225  244  253    …      151
0
 [8,] 0    0    0    0    0    0    0   30  149   78    …      224
0
 [9,] 0    0    0    0    0    0    0    0    0    0    …      224
0
[10,] 0    0    0    0    0    0    0    0    0    0    …      224
0
[11,] 0    0    0    0    0    0    0    0    0    0    …      224
0
[12,] 0    0    0    0    0    0    0    0    0    0    …      240
121
[13,] 0    0    0    0    0    0    0    0    0    0    …      253
253
[14,] 0    0    0    0    0    0    0    0    0    0    …       97
253
[15,] 0    0    0    0    0    0    0    0    0    0    …      122
253
[16,] 0    0    0    0    0    0    0    0    0    0    …      237
253
[17,] 0    0    0    0    0    0    0    0    0    0    …      253
253
[18,] 0    0    0    0    0    0    0    0    0    0    …      253
253
[19,] 0    0    0    0    0    0    0    0    0    0    …      253
212
[20,] 0    0    0    0    0    0   33  136   70    6    …      234
```

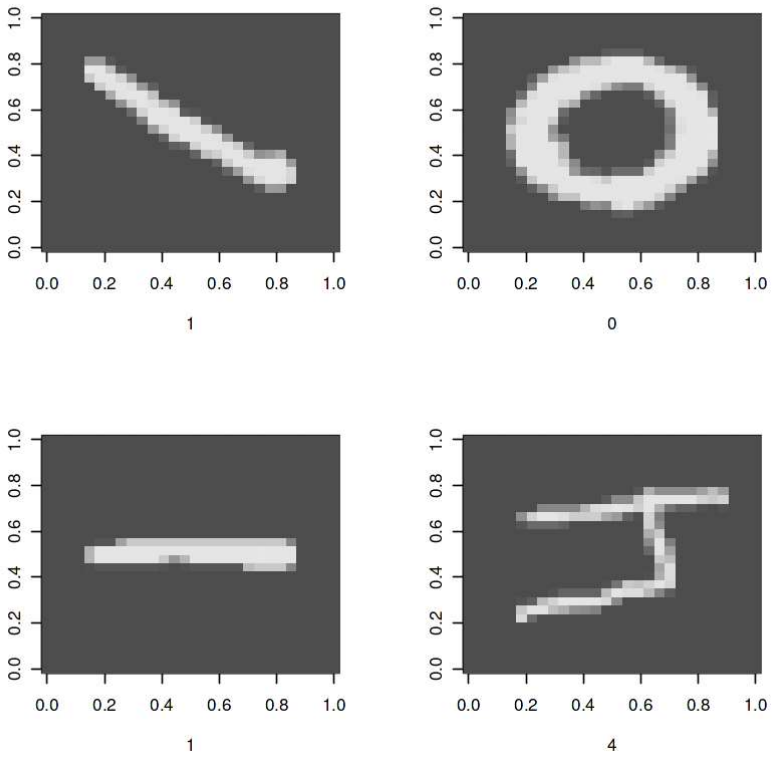
```
31
[21,] 0    0    0    0    0    26   231  253  253  191  …     112
0
[22,] 0    0    0    0    0    36   215  253  253  253  …       0
0
[23,] 0    0    0    0    0     5    87  223  253  253  …       0
0
[24,] 0    0    0    0    0     0    67   50  176  148  …       0
0
[25,] 0    0    0    0    0     0     0    0    0    0  …       0
0
[26,] 0    0    0    0    0     0     0    0    0    0  …       0
0
[27,] 0    0    0    0    0     0     0    0    0    0  …       0
0
[28,] 0    0    0    0    0     0     0    0    0    0  …       0
0
      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21]
 [1,]   0     0   0     0     0     0     0     0
 [2,]   0     0   0     0     0     0     0     0
 [3,]   0     0   0     0     0     0     0     0
 [4,]   0     0   0     0     0     0     0     0
 [5,]   0     0   0     0     0     0     0     0
 [6,]   0     0   0     0     0     0     0     0
 [7,]   0     0   0     0     0     0     0     0
 [8,]   0     0   0     0     0     0     0     0
 [9,]   0     0   0     0     0     0     0     0
[10,]   0     0   0     0     0     0     0     0
[11,]   0     0   0     0     0     0     0     0
[12,]   7     0   0     0     0     0     0     0
[13,] 185    53   0     0     0     0     0     0
[14,] 253   170   0     0     0     0     0     0
[15,] 253   170   0     0     0     0     0     0
[16,] 253   170   0     0     0     0     0     0
[17,] 253   170   0     0     0     0     0     0
[18,] 214    28   0     0     0     0     0     0
[19,]  30     0   0     0     0     0     0     0
[20,]   0     0   0     0     0     0     0     0
[21,]   0     0   0     0     0     0     0     0
[22,]   0     0   0     0     0     0     0     0
[23,]   0     0   0     0     0     0     0     0
[24,]   0     0   0     0     0     0     0     0
[25,]   0     0   0     0     0     0     0     0
[26,]   0     0   0     0     0     0     0     0
[27,]   0     0   0     0     0     0     0     0
[28,]   0     0   0     0     0     0     0     0
```




In the image and matrix above, we can see that the highest pixelated values are centered in the middle of the matrix, just as the digit is centered in the middle of the image. Values that record blank space are noted as 0s. However, the orientation of the digit is sideways. Do the remaining images have random orientation or is it common?

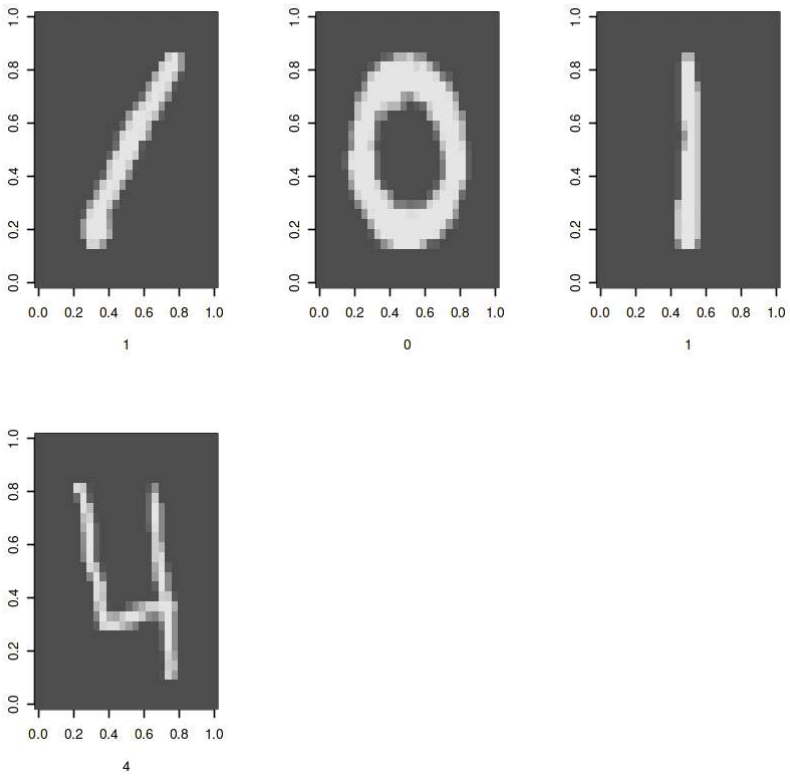
```
par(mfrow=c(2,2)) #the larger the number, the smaller the graph
lapply(1:4,
    function(x) image(matrix(unlist(train_data[x,-1]),nrow = 28,byrow
= T),
                      col=grey.colors(255),
                      xlab=train_data[x,1]
                  )
)
```

```
[[1]]
NULL

[[2]]
NULL

[[3]]
NULL

[[4]]
NULL
```




The orientation of the hand-written images appear to be somewhat sideways, so they should all be upright for the visualization and for machine learning.

```
rotate <- function(x) t(apply(x, 2, rev)) # reverses (rotates the
matrix)
par(mfrow=c(2,3))
lapply(1:4,
    function(x) image(rotate(matrix(unlist(train_data[x,-1]),nrow =
28,byrow = T)),
                      col=grey.colors(255),
                      xlab=train_data[x,1]
                  )
)

[[1]]
NULL

[[2]]
NULL

[[3]]
NULL

[[4]]
NULL
```




Subset training data by label digit and display a few of each.

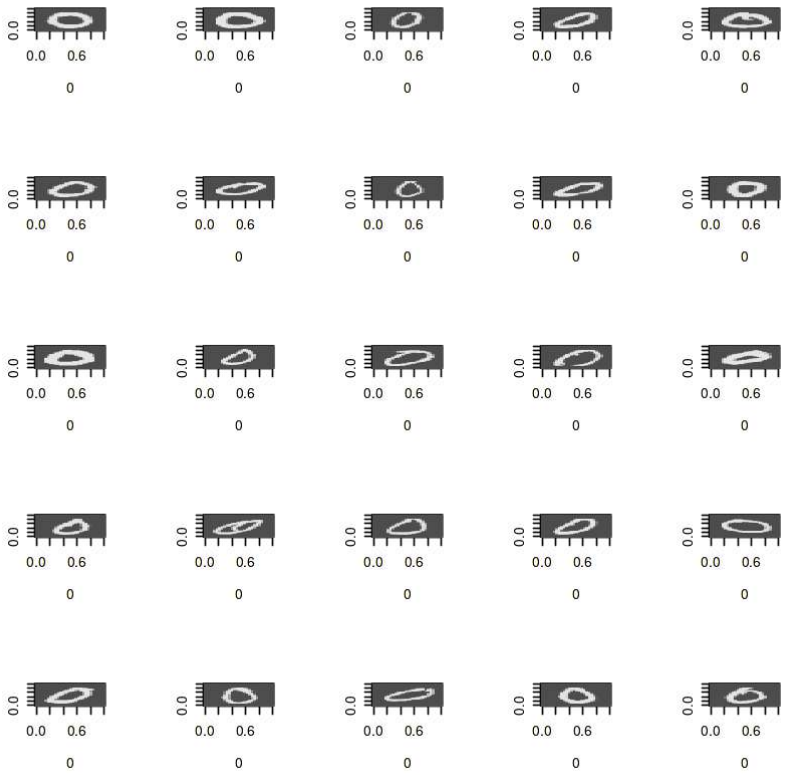
```
### library(dplyr)
grouped_0 <- train_data %>%
    subset(train_data$label== 0)
head(grouped_0)

par(mfrow=c(5,5))
lapply(1:28,
    function(x) image(rotate(matrix(unlist(grouped_0[x,-1]),nrow =
28,byrow = T)),
                      col=grey.colors(255),
                      xlab=grouped_0[x,1]
                  )
)
```

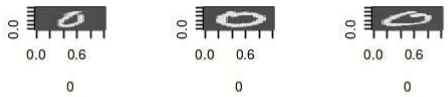
| | label | pixel0 | pixel1 | pixel2 | pixel3 | pixel4 | pixel5 | pixel6 | pixel7 | pixel8 |
|---|---|---|---|---|---|---|---|---|---|---|
| … | | | | | | | | | | |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| … | | | | | | | | | | |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| … | | | | | | | | | | |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| … | | | | | | | | | | |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

```
…
5 0     0      0      0      0      0      0      0      0      0
…
6 0     0      0      0      0      0      0      0      0      0
…
  pixel774 pixel775 pixel776 pixel777 pixel778 pixel779 pixel780
pixel781
1 0        0        0        0        0        0        0        0

2 0        0        0        0        0        0        0        0

3 0        0        0        0        0        0        0        0

4 0        0        0        0        0        0        0        0

5 0        0        0        0        0        0        0        0

6 0        0        0        0        0        0        0        0

  pixel782 pixel783
1 0        0
2 0        0
3 0        0
4 0        0
5 0        0
6 0        0

[[1]]
NULL

[[2]]
NULL

[[3]]
NULL

[[4]]
NULL

[[5]]
NULL

[[6]]
NULL

[[7]]
NULL

[[8]]
NULL
```

```
[[9]]
NULL

[[10]]
NULL

[[11]]
NULL

[[12]]
NULL

[[13]]
NULL

[[14]]
NULL

[[15]]
NULL

[[16]]
NULL

[[17]]
NULL

[[18]]
NULL

[[19]]
NULL

[[20]]
NULL

[[21]]
NULL

[[22]]
NULL

[[23]]
NULL

[[24]]
NULL

[[25]]
NULL
```

```
[[26]]
NULL

[[27]]
NULL

[[28]]
NULL
```






```
grouped_digits <- train_data %>%
    group by(train_data$label) %>% slice(1:2)
(grouped_digits)
```

```
   label pixel0 pixel1 pixel2 pixel3 pixel4 pixel5 pixel6 pixel7
pixel8 …
1  0     0      0      0      0      0      0      0      0      0
…
2  0     0      0      0      0      0      0      0      0      0
…
3  1     0      0      0      0      0      0      0      0      0
…
4  1     0      0      0      0      0      0      0      0      0
…
5  2     0      0      0      0      0      0      0      0      0
…
6  2     0      0      0      0      0      0      0      0      0
…
7  3     0      0      0      0      0      0      0      0      0
…
8  3     0      0      0      0      0      0      0      0      0
…
9  4     0      0      0      0      0      0      0      0      0
…
```

```
10 4     0      0      0      0      0      0      0      0      0
…
11 5     0      0      0      0      0      0      0      0      0
…
12 5     0      0      0      0      0      0      0      0      0
…
13 6     0      0      0      0      0      0      0      0      0
…
14 6     0      0      0      0      0      0      0      0      0
…
15 7     0      0      0      0      0      0      0      0      0
…
16 7     0      0      0      0      0      0      0      0      0
…
17 8     0      0      0      0      0      0      0      0      0
…
18 8     0      0      0      0      0      0      0      0      0
…
19 9     0      0      0      0      0      0      0      0      0
…
20 9     0      0      0      0      0      0      0      0      0
…
   pixel775 pixel776 pixel777 pixel778 pixel779 pixel780 pixel781
pixel782
1  0        0        0        0        0        0        0         0

2  0        0        0        0        0        0        0         0

3  0        0        0        0        0        0        0         0

4  0        0        0        0        0        0        0         0

5  0        0        0        0        0        0        0         0

6  0        0        0        0        0        0        0         0

7  0        0        0        0        0        0        0         0

8  0        0        0        0        0        0        0         0

9  0        0        0        0        0        0        0         0

10 0        0        0        0        0        0        0         0

11 0        0        0        0        0        0        0         0

12 0        0        0        0        0        0        0         0

13 0        0        0        0        0        0        0         0
```

```
14 0        0        0        0        0        0        0        0

15 0        0        0        0        0        0        0        0

16 0        0        0        0        0        0        0        0

17 0        0        0        0        0        0        0        0

18 0        0        0        0        0        0        0        0

19 0        0        0        0        0        0        0        0

20 0        0        0        0        0        0        0        0
```

```
   pixel783 train_data$label
1  0        0
2  0        0
3  0        1
4  0        1
5  0        2
6  0        2
7  0        3
8  0        3
9  0        4
10 0        4
11 0        5
12 0        5
13 0        6
14 0        6
15 0        7
16 0        7
17 0        8
18 0        8
19 0        9
20 0        9
```

```
#https://www.geeksforgeeks.org/cross-validation-in-r-programming/
#library(caret)
#library(tidyverse)
#library(e1071)
library(randomForest)
set.seed(1)
#The x data are the contributing attributes while the y is the feature
the model is trying to determine
x=train_data[,2:785]
y=train_data$label
```

```
randomForest 4.6-14

Type rfNews() to see new features/changes/bug fixes.
```

```
Attaching package: ‘randomForest’

The following object is masked from ‘package:dplyr’:

    combine

The following object is masked from ‘package:ggplot2’:

    margin
```

```
#xg boost
require(xgboost)
require(Matrix)
require(data.table)
if (!require('vcd')) install.packages('vcd')
```

```
Loading required package: xgboost

Attaching package: ‘xgboost’

The following object is masked from ‘package:dplyr’:

    slice

Loading required package: Matrix

Attaching package: ‘Matrix’

The following objects are masked from ‘package:tidyr’:

    expand, pack, unpack

Loading required package: data.table

Attaching package: ‘data.table’

The following objects are masked from ‘package:lubridate’:

    hour, isoweek, mday, minute, month, quarter, second, wday, week,
```

```
    yday, year

The following objects are masked from 'package:dplyr':

    between, first, last

The following object is masked from 'package:purrr':

    transpose

Loading required package: vcd

Loading required package: grid
```

```
#nrounds max number of boosting iterations.
#max_depth maximum depth of the tree
library(caTools)
split <- sample.split(train_data$label, SplitRatio = 0.75)
train_subset <- subset(train_data, split == T)
nrow(train_data)
nrow(train_subset)
#ensure all digits are represented, otherwise there will be errors in
learning
table(train_subset$label)
val_subset <- subset(train_data, split == F)
nrow(val_subset)
#head(cv)
train_matrix <- as.data.frame(lapply(train_subset, as.numeric))
#cv <- as.data.frame(lapply(cv, as.numeric))
train_matrix_formatted <- xgb.DMatrix(data =
data.matrix(train_subset[,-1]), label = train_subset$label)
test_matrix <- xgb.DMatrix(data = data.matrix(val_subset[,-1]), label
= val_subset$label)

#watchlist is provided so that the model has data to train and
validation data to test its learning while the model is made
watch_list <- list(train = train_matrix_formatted, test = test_matrix)
```

```
[1] 42000

[1] 31500

   0    1    2    3    4    5    6    7    8    9
3099 3513 3133 3263 3054 2846 3103 3301 3047 3141

[1] 10500
```

```
trained_model <- xgb.train(data = train_matrix_formatted,   max_depth
= 20, eta = 0.5,
                   early_stopping_rounds=5, nthread = 2, nrounds = 10,
objective = "multi:softmax", num_class=10
                    ,eval_metric = "merror", watchlist=watch_list)
```

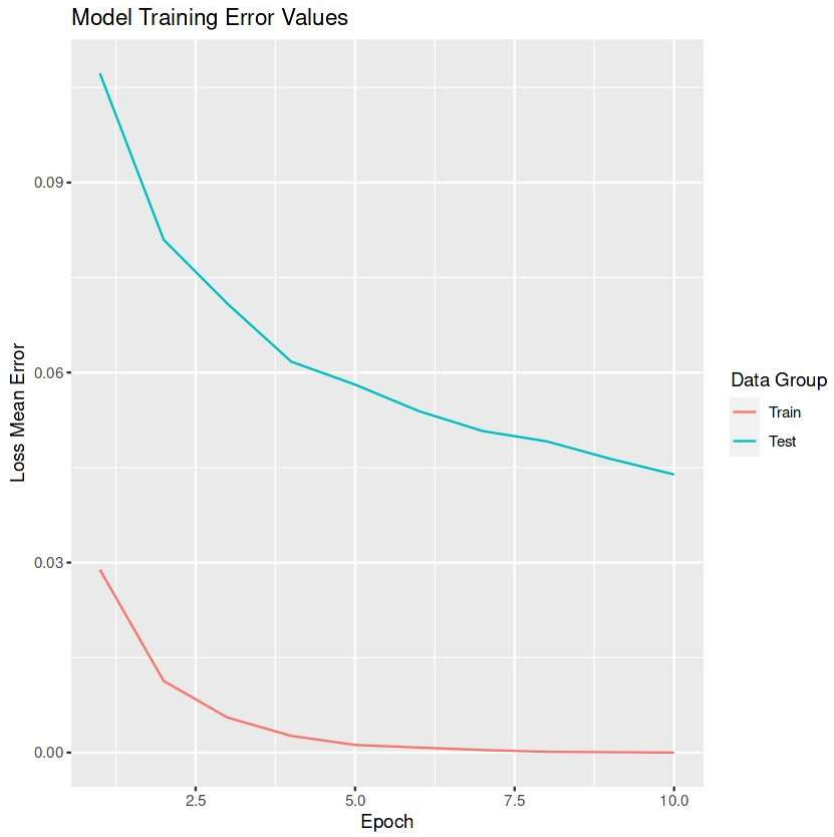
```
[1]  train-merror:0.028889 test-merror:0.107238
Multiple eval metrics are present. Will use test_merror for early
stopping.
Will train until test_merror hasn't improved in 5 rounds.

[2]  train-merror:0.011302 test-merror:0.080952
[3]  train-merror:0.005524 test-merror:0.070857
[4]  train-merror:0.002635 test-merror:0.061714
[5]  train-merror:0.001206 test-merror:0.058095
[6]  train-merror:0.000794 test-merror:0.053905
[7]  train-merror:0.000413 test-merror:0.050762
[8]  train-merror:0.000127 test-merror:0.049143
[9]  train-merror:0.000063 test-merror:0.046381
[10] train-merror:0.000000 test-merror:0.043905
```

```
str(trained_model)
library(ggplot2)
#https://stackoverflow.com/questions/44589103/xgboost-r-cv-test-vs-
training-error
plot_error <- data.frame()
plot_structure<-
data.frame(Epoch=as.matrix(trained_model$evaluation_log)[,1],
Train=as.matrix(trained_model$evaluation_log)[,2],
Test=as.matrix(trained_model$evaluation_log)[,3])
#reshape into long format so it can be graphed by group
library(reshape)
long_structure <-melt(plot_structure, measure.vars=2:3)
plot_model <- ggplot(data=long_structure, aes(x=Epoch, y=value,
colour=variable)) + geom_line() + ylab("Loss Mean Error")+
ggtitle("Model Training Error Values")
plot_model1 <- plot_model+ guides(color = guide_legend(title = "Data
Group"))
plot_model1
```

```
List of 13
 $ handle         :Class 'xgb.Booster.handle' <externalptr>
 $ raw            : raw [1:1299422] 7b 4c 00 00 ...
 $ best_iteration : num 10
 $ best_ntreelimit: int 10
 $ best_score     : num 0.0439
 $ best_msg       : chr "[10]\ttrain-merror:0.000000\ttest-
merror:0.043905"
 $ niter          : num 10
 $ evaluation_log :Classes ‘data.table’ and 'data.frame':   10 obs. of
```

```
3 variables:
  ..$ iter        : num [1:10] 1 2 3 4 5 6 7 8 9 10
  ..$ train_merror: num [1:10] 0.02889 0.0113 0.00552 0.00263
0.00121 ...
  ..$ test_merror : num [1:10] 0.1072 0.081 0.0709 0.0617 0.0581 ...
  ..- attr(*, ".internal.selfref")=<externalptr>
 $ call           : language xgb.train(data = train_matrix_formatted,
nrounds = 10, watchlist = watch_list,      early_stopping_rounds = 5,
ma| __truncated__ ...
 $ params         :List of 7
  ..$ max_depth          : num 20
  ..$ eta                : num 0.5
  ..$ nthread            : num 2
  ..$ objective          : chr "multi:softmax"
  ..$ num_class          : num 10
  ..$ eval_metric        : chr "merror"
  ..$ validate_parameters: logi TRUE
 $ callbacks      :List of 3
  ..$ cb.print.evaluation:function (env = parent.frame())
  .. ..- attr(*, "call")= language cb.print.evaluation(period =
print_every_n)
  .. ..- attr(*, "name")= chr "cb.print.evaluation"
  ..$ cb.evaluation.log  :function (env = parent.frame(), finalize =
FALSE)
  .. ..- attr(*, "call")= language cb.evaluation.log()
  .. ..- attr(*, "name")= chr "cb.evaluation.log"
  ..$ cb.early.stop      :function (env = parent.frame(), finalize =
FALSE)
  .. ..- attr(*, "call")= language cb.early.stop(stopping_rounds =
early_stopping_rounds, maximize = maximize,      verbose = verbose)
  .. ..- attr(*, "name")= chr "cb.early.stop"
 $ feature_names  : chr [1:784] "pixel0" "pixel1" "pixel2"
"pixel3" ...
 $ nfeatures      : int 784
 - attr(*, "class")= chr "xgb.Booster"


Attaching package: ‘reshape’


The following object is masked from ‘package:data.table’:

    melt


The following object is masked from ‘package:Matrix’:

    expand
```

```
The following object is masked from ‘package:lubridate’:

    stamp

The following object is masked from ‘package:dplyr’:

    rename

The following objects are masked from ‘package:tidyr’:

    expand, smiths
```




The large gap between the training and test error lines tell me that the model is learning alot of noise. Its parameters need to be finetuned. Training loss decreases as the model learns any data available. The validation loss decreases when the model learns signal or any revelant patterns for prediction. The aim is to find a medium group where the validation loss does not increase. This is why in the model used there is a paramter called "early_stopping_rounds". This value tells the model to stop training when there is no significant decrease in the validation loss for 5 epochs. Let's finetune the model.

```
new_model_nrounds <- xgb.train(data = train_matrix_formatted,
max_depth = 20, eta = 0.5,
                   early_stopping_rounds=5, nthread = 2, nrounds = 20,
objective = "multi:softmax", num_class=10
                    ,eval_metric = "merror", watchlist=watch_list)
```

```
[1]  train-merror:0.028889 test-merror:0.107238
Multiple eval metrics are present. Will use test_merror for early
stopping.
Will train until test_merror hasn't improved in 5 rounds.

[2]  train-merror:0.011302 test-merror:0.080952
[3]  train-merror:0.005524 test-merror:0.070857
[4]  train-merror:0.002635 test-merror:0.061714
[5]  train-merror:0.001206 test-merror:0.058095
[6]  train-merror:0.000794 test-merror:0.053905
[7]  train-merror:0.000413 test-merror:0.050762
[8]  train-merror:0.000127 test-merror:0.049143
[9]  train-merror:0.000063 test-merror:0.046381
[10] train-merror:0.000000 test-merror:0.043905
[11] train-merror:0.000000 test-merror:0.043810
[12] train-merror:0.000000 test-merror:0.042762
[13] train-merror:0.000000 test-merror:0.041238
[14] train-merror:0.000000 test-merror:0.040286
[15] train-merror:0.000000 test-merror:0.039619
[16] train-merror:0.000000 test-merror:0.038857
[17] train-merror:0.000000 test-merror:0.038095
[18] train-merror:0.000000 test-merror:0.036762
[19] train-merror:0.000000 test-merror:0.036476
[20] train-merror:0.000000 test-merror:0.035810
```

With more rounds, the test error has only decreased slightly.

```
new_model_depth <- xgb.train(data = train_matrix_formatted,
max_depth = 30, eta = 0.5,
                   early_stopping_rounds=5, nthread = 2, nrounds = 20,
objective = "multi:softmax", num_class=10
                    ,eval_metric = "merror", watchlist=watch_list)
```

```
[1]  train-merror:0.026730 test-merror:0.107619
Multiple eval metrics are present. Will use test_merror for early
stopping.
Will train until test_merror hasn't improved in 5 rounds.

[2]  train-merror:0.010317 test-merror:0.079143
[3]  train-merror:0.005238 test-merror:0.068952
[4]  train-merror:0.002571 test-merror:0.061905
[5]  train-merror:0.001238 test-merror:0.056286
[6]  train-merror:0.000667 test-merror:0.052286
[7]  train-merror:0.000349 test-merror:0.049429
```

```
[8]   train-merror:0.000127 test-merror:0.048000
[9]   train-merror:0.000063 test-merror:0.046476
[10]  train-merror:0.000000 test-merror:0.044571
[11]  train-merror:0.000000 test-merror:0.043429
[12]  train-merror:0.000000 test-merror:0.041429
[13]  train-merror:0.000000 test-merror:0.040952
[14]  train-merror:0.000000 test-merror:0.039714
[15]  train-merror:0.000000 test-merror:0.039524
[16]  train-merror:0.000000 test-merror:0.038952
[17]  train-merror:0.000000 test-merror:0.037714
[18]  train-merror:0.000000 test-merror:0.037143
[19]  train-merror:0.000000 test-merror:0.036571
[20]  train-merror:0.000000 test-merror:0.035619
```

Increasing the max depth of a tree improved the model slightly.

```
new_model_depth_rounds <- xgb.train(data = train_matrix_formatted,
max_depth = 30, eta = 0.5,
                   early_stopping_rounds=5, nthread = 2, nrounds = 50,
objective = "multi:softmax", num_class=10
                    ,eval_metric = "merror", watchlist=watch_list)
```

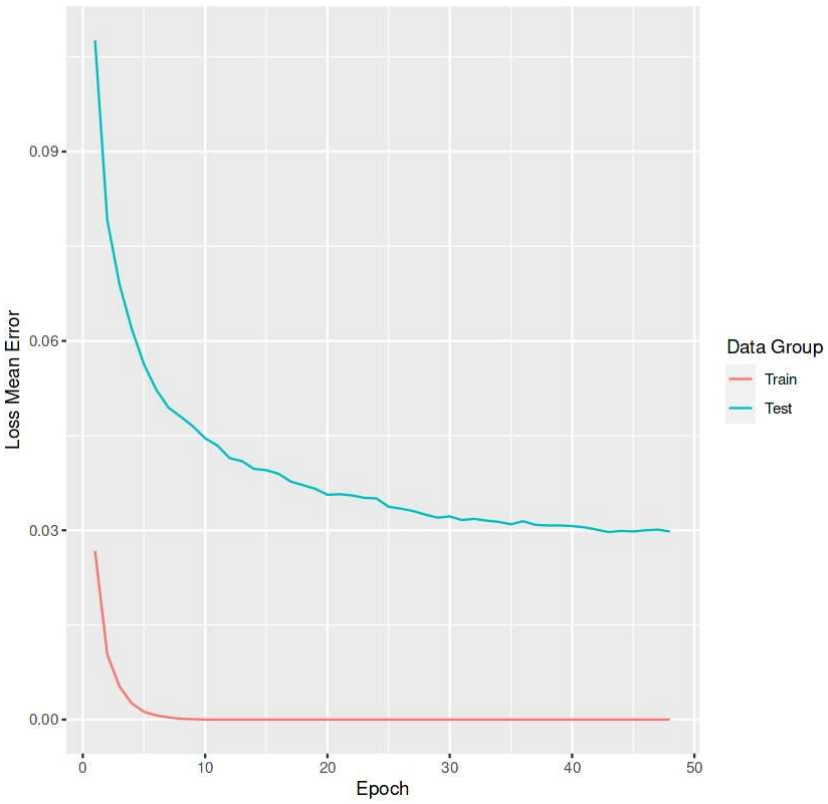
```
[1]   train-merror:0.026730 test-merror:0.107619
Multiple eval metrics are present. Will use test_merror for early
stopping.
Will train until test_merror hasn't improved in 5 rounds.

[2]   train-merror:0.010317 test-merror:0.079143
[3]   train-merror:0.005238 test-merror:0.068952
[4]   train-merror:0.002571 test-merror:0.061905
[5]   train-merror:0.001238 test-merror:0.056286
[6]   train-merror:0.000667 test-merror:0.052286
[7]   train-merror:0.000349 test-merror:0.049429
[8]   train-merror:0.000127 test-merror:0.048000
[9]   train-merror:0.000063 test-merror:0.046476
[10]  train-merror:0.000000 test-merror:0.044571
[11]  train-merror:0.000000 test-merror:0.043429
[12]  train-merror:0.000000 test-merror:0.041429
[13]  train-merror:0.000000 test-merror:0.040952
[14]  train-merror:0.000000 test-merror:0.039714
[15]  train-merror:0.000000 test-merror:0.039524
[16]  train-merror:0.000000 test-merror:0.038952
[17]  train-merror:0.000000 test-merror:0.037714
[18]  train-merror:0.000000 test-merror:0.037143
[19]  train-merror:0.000000 test-merror:0.036571
[20]  train-merror:0.000000 test-merror:0.035619
[21]  train-merror:0.000000 test-merror:0.035714
[22]  train-merror:0.000000 test-merror:0.035524
[23]  train-merror:0.000000 test-merror:0.035143
```

```
[24]  train-merror:0.000000 test-merror:0.035048
[25]  train-merror:0.000000 test-merror:0.033714
[26]  train-merror:0.000000 test-merror:0.033429
[27]  train-merror:0.000000 test-merror:0.033048
[28]  train-merror:0.000000 test-merror:0.032476
[29]  train-merror:0.000000 test-merror:0.032000
[30]  train-merror:0.000000 test-merror:0.032190
[31]  train-merror:0.000000 test-merror:0.031619
[32]  train-merror:0.000000 test-merror:0.031810
[33]  train-merror:0.000000 test-merror:0.031524
[34]  train-merror:0.000000 test-merror:0.031333
[35]  train-merror:0.000000 test-merror:0.030952
[36]  train-merror:0.000000 test-merror:0.031429
[37]  train-merror:0.000000 test-merror:0.030857
[38]  train-merror:0.000000 test-merror:0.030762
[39]  train-merror:0.000000 test-merror:0.030762
[40]  train-merror:0.000000 test-merror:0.030667
[41]  train-merror:0.000000 test-merror:0.030476
[42]  train-merror:0.000000 test-merror:0.030095
[43]  train-merror:0.000000 test-merror:0.029714
[44]  train-merror:0.000000 test-merror:0.029905
[45]  train-merror:0.000000 test-merror:0.029810
[46]  train-merror:0.000000 test-merror:0.030000
[47]  train-merror:0.000000 test-merror:0.030095
[48]  train-merror:0.000000 test-merror:0.029810
Stopping. Best iteration:
[43]  train-merror:0.000000 test-merror:0.029714
```

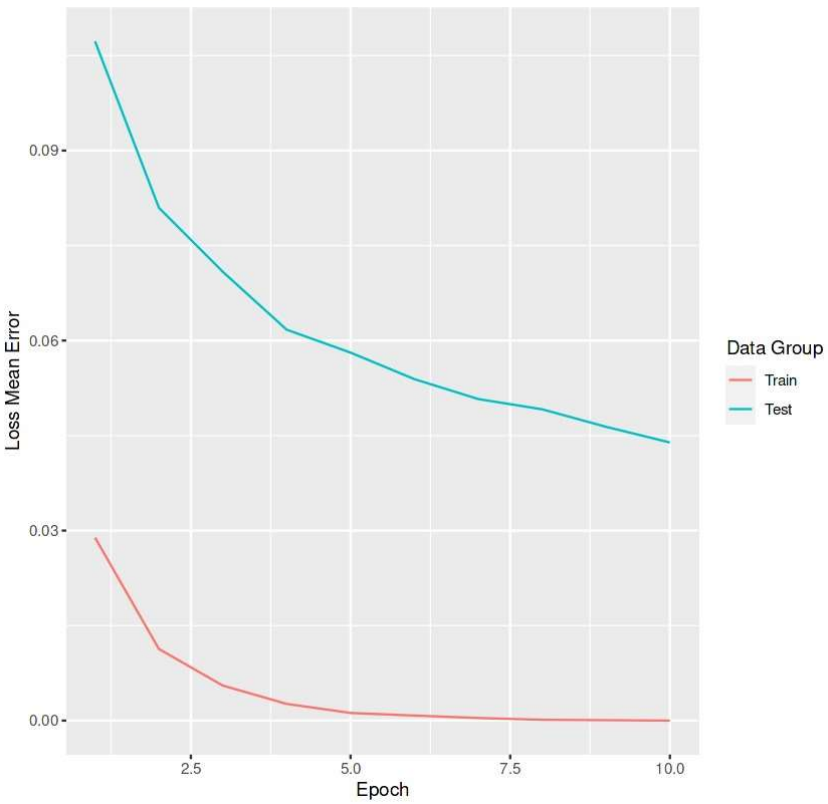
The previous model shows better improvement. Adding more epochs allows the model to train longer and exemplifies the early stopping paramter.

```
plot_structure_new<-
data.frame(Epoch=as.matrix(new_model_depth_rounds$evaluation_log)[,1],
Train=as.matrix(new_model_depth_rounds$evaluation_log)[,2],
Test=as.matrix(new_model_depth_rounds$evaluation_log)[,3])
#reshape into long format so it can be graphed by group
library(reshape)
long_structure_new <-melt(plot_structure_new, measure.vars=2:3)
plot_model_new <- ggplot(data=long_structure_new, aes(x=Epoch,
y=value, colour=variable)) + geom_line() + ylab("Loss Mean Error")+
ggtitle("Improved Model Training Error Values")
plot_model_improved <- plot_model_new+ guides(color =
guide_legend(title = "Data Group"))
plot_model_improved
plot_model1
```

Improved Model Training Error Values




Model Training Error Values



```
#https://rpubs.com/fsmithus/digits-xgboost
#str(new_model_depth_rounds)
test_formatted <- xgb.DMatrix(data = data.matrix(test_data))
output <- predict(new_model_depth_rounds,(test_formatted))
str(output)
output_labels <- as.data.frame(output)
head(output_labels)
names(output_labels)[1]= "Label"
results <- data.frame(ImageId= 1:nrow(output_labels), Label=
output_labels)

write.csv(results, '/kaggle/working/submission.csv', row.names= FALSE)
```

```
 num [1:28000] 2 0 9 9 2 7 0 3 0 3 ...
```

```
  output
1 2
2 0
3 9
4 9
5 2
6 7
```