Final Project – Digital Handwriting Classification

Brian Mallari

Introduction

Two data sets of pixel data for greyscale images of hand-written digits have been provided. The data sets come from MNIST (or the Modified National Institute of Standards and Technology), and they serve as a reliable benchmark for classification algorithms. The digits in each image range from 0 to 9 inclusive. The images themselves are each 28 pixels high and 28 pixels wide; altogether, each image consists of 784 pixels.

The first data set ("train.csv") starts with a column titled "label" which indicates what the hand-written digit is. Next are an additional 784 columns each consisting of pixel values that range from 0 to 255 inclusive, with higher values corresponding to darker pixels. Altogether, there are 785 columns in the first data set.

For the columns relating to the pixels, each column is titled "labelx" where the second x is a number generated from the following formula,

$$x = (i * 28) + j$$

where i is the row position (indexed from 0) of the pixel within a given image, and j is the column position (also indexed from 0) of the same pixel within the same image, with both i and j ranging from 0 to 27 inclusive. Therefore, a value under "label0" corresponds to the pixel located in the first row from the top of the image (i = 0) and the first column from the left of the same image (j = 0). Likewise, a value under "label783" corresponds to the pixel located in the last row from the top of the image (i = 27) and the last column from the left of the same image (j = 27).

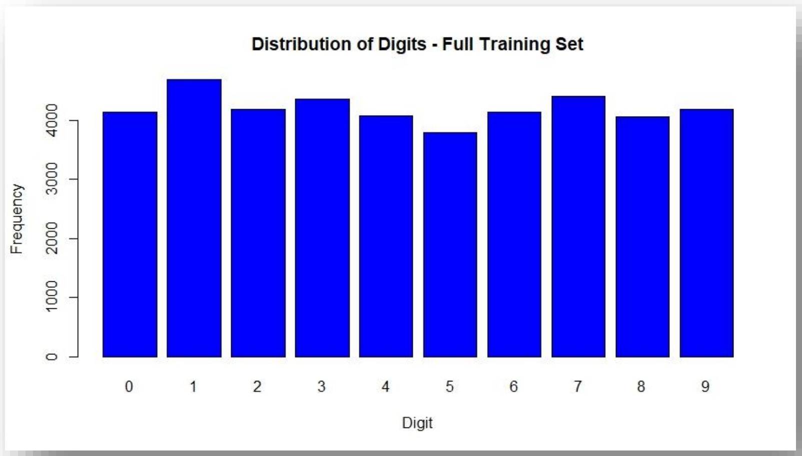
The second data set ("test.csv") follows the same format as the first data set except that there is no "label" column. Altogether, the second data set has 784 columns.

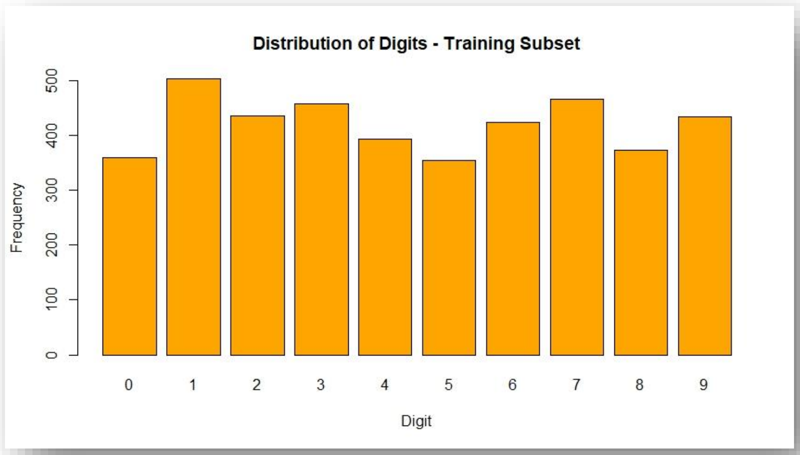
The objective here is to utilize the data set of labeled images to generate classification models that can relate the pixel data of an image to its label, and then apply the most accurate of those models to the second data set in order to assign a label to an image based solely on the pixel data that particular image.

Data Wrangling

The first data set with labels ("train.csv") consists of 42,000 observations each with 785 dimensions. Due to the sheer size of this data set, a subset of this training set consisting of 4,200 observations (or 10% of the full data set) was utilized for generating the classification models.

Here are the distributions of the labels for both the full training set and the training subset:





The two distributions of digits look similar, so any analysis performed on the training subset is expected to produce results comparable to the same analysis preformed on the full data set.

Moreover, a second, modified version of the training subset was generated where all pixel values above 0 was replaced with 1. The rationale here was that the darkness of the pixel wasn't as important as whether or not there's even a pixel present at all. This "flattening" of the pixel values is akin to removing the greyscale and replacing it with strictly a solid shade, and the intention is to 1.) apply as many classification techniques as possible, and to 2.) potentially increase the accuracy of some of the classification techniques. This modified version of the training subset was saved as a separate .csv file ("digit_train_subset_mod_pixel_off_or_on.csv") for easy access, since each transformation process of the numerical training subset can take up to several hours depending on the computer hardware.

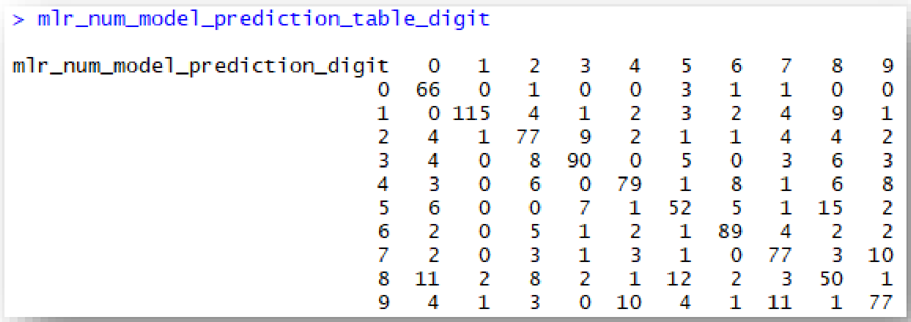
The labels for both the first, numerical training subset and the second, binary training subset were converted to factor levels to enforce the fact that the label values are really a description of what the image represents and not a value that can be subject to mathematical operations, such as addition or multiplication. The pixel values of the numeric training set were kept as integers, while the pixel values for the binary training set were converted to factor levels.

Both the numeric and binary training sets were each split into two subsets – one set with 75% of the observations for building the classification models, and one with the remaining 25% of the data for evaluating the classification models. Neither the molding-building subsets nor the model-evaluation subsets will be identical due to pseudo-randomized nature of the data split for the numeric and binary subsets. The version of the training subset used for building a classification model will be explained for each model that was generated.

Classification Models

Multinomial Logistic Regression (MLR)

Here is a table comparing the labels predicted with the numeric pixel values of the model evaluation subset and the actual labels of the same subset:



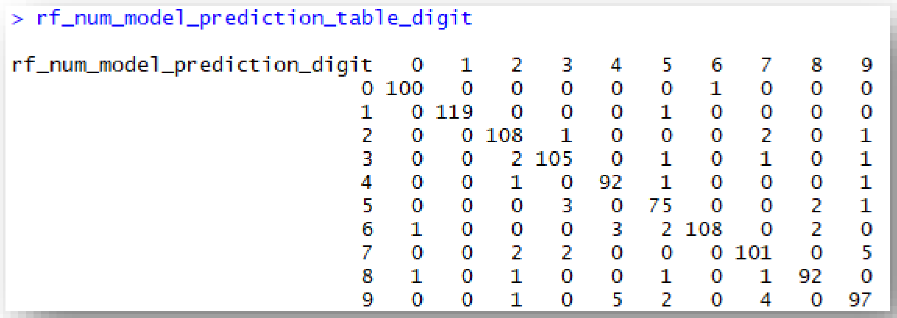
```
> mlr_num_model_prediction_table_digit

mlr_num_model_prediction_digit   0   1   2   3   4   5   6   7   8   9
                             0  66   0   1   0   0   3   1   1   0   0
                             1   0 115   4   1   2   3   2   4   9   1
                             2   4   1  77   9   2   1   1   4   4   2
                             3   4   0   8  90   0   5   0   3   6   3
                             4   3   0   6   0  79   1   8   1   6   8
                             5   6   0   0   7   1  52   5   1  15   2
                             6   2   0   5   1   2   1  89   4   2   2
                             7   2   0   3   1   3   1   0  77   3  10
                             8  11   2   8   2   1  12   2   3  50   1
                             9   4   1   3   0  10   4   1  11   1  77
```

Accuracy = 0.7352381

A model using the binary version of the pixel values could not be generated because the function for generating the MLR model requires factors with at least two levels for the sake of contrast, and the factors with just 0 (i.e. no pixels present) had only one level.

Random Forest (RF)

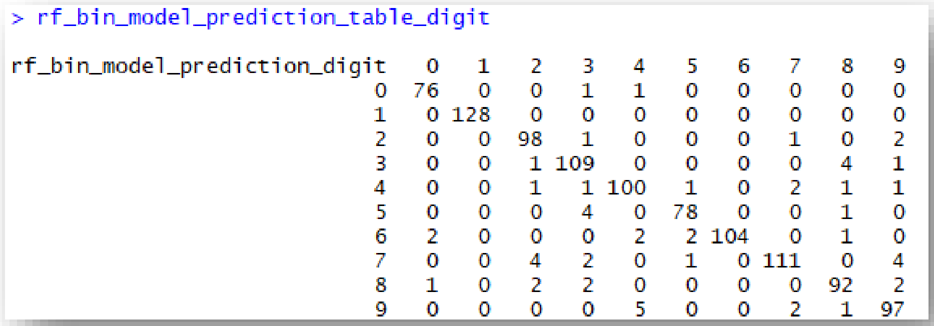
Here is a table comparing the labels predicted with the numeric pixel values of the model evaluation subset and the actual labels of the same subset:



```
> rf_num_model_prediction_table_digit

rf_num_model_prediction_digit   0   1   2   3   4   5   6   7   8   9
                            0 100   0   0   0   0   0   1   0   0   0
                            1   0 119   0   0   0   1   0   0   0   0
                            2   0   0 108   1   0   0   0   2   0   1
                            3   0   0   2 105   0   1   0   1   0   1
                            4   0   0   1   0  92   1   0   0   0   1
                            5   0   0   0   3   0  75   0   0   2   1
                            6   1   0   0   0   3   2 108   0   2   0
                            7   0   0   2   2   0   0   0 101   0   5
                            8   1   0   1   0   0   1   0   1  92   0
                            9   0   0   1   0   5   2   0   4   0  97
```

Accuracy = 0.9495238

Here is a table comparing the labels predicted with the binary pixel values of the model evaluation subset and the actual labels of the same subset:

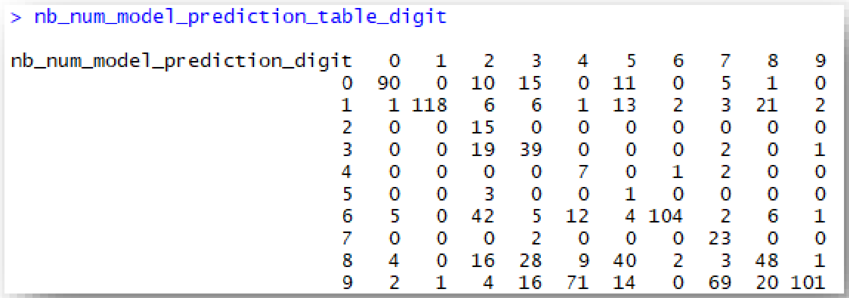


```
> rf_bin_model_prediction_table_digit

rf_bin_model_prediction_digit   0   1   2   3   4   5   6   7   8   9
                            0  76   0   0   1   1   0   0   0   0   0
                            1   0 128   0   0   0   0   0   0   0   0
                            2   0   0  98   1   0   0   0   1   0   2
                            3   0   0   1 109   0   0   0   0   4   1
                            4   0   0   1   1 100   1   0   2   1   1
                            5   0   0   0   4   0  78   0   0   1   0
                            6   2   0   0   0   2   2 104   0   1   0
                            7   0   0   4   2   0   1   0 111   0   4
                            8   1   0   2   2   0   0   0   0  92   2
                            9   0   0   0   0   5   0   0   2   1  97
```

Accuracy = 0.9457143

Naïve Bayes (NB)

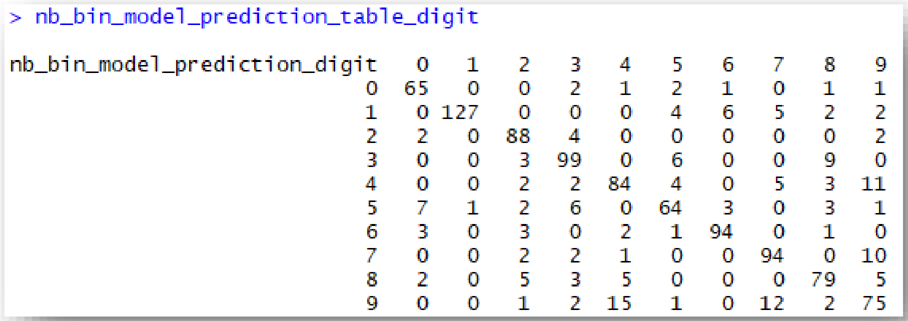
Here is a table comparing the labels predicted with the numeric pixel values of the model evaluation subset and the actual labels of the same subset:



```
> nb_num_model_prediction_table_digit

nb_num_model_prediction_digit   0   1   2   3   4   5   6   7   8   9
                            0  90   0  10  15   0  11   0   5   1   0
                            1   1 118   6   6   1  13   2   3  21   2
                            2   0   0  15   0   0   0   0   0   0   0
                            3   0   0  19  39   0   0   0   2   0   1
                            4   0   0   0   0   7   0   1   2   0   0
                            5   0   0   3   0   0   1   0   0   0   0
                            6   5   0  42   5  12   4 104   2   6   1
                            7   0   0   0   2   0   0   0  23   0   0
                            8   4   0  16  28   9  40   2   3  48   1
                            9   2   1   4  16  71  14   0  69  20 101
```

Accuracy = 0.52

Here is a table comparing the labels predicted with the binary pixel values of the model evaluation subset and the actual labels of the same subset:



```
> nb_bin_model_prediction_table_digit

nb_bin_model_prediction_digit   0   1   2   3   4   5   6   7   8   9
                            0  65   0   0   2   1   2   1   0   1   1
                            1   0 127   0   0   0   4   6   5   2   2
                            2   2   0  88   4   0   0   0   0   0   2
                            3   0   0   3  99   0   6   0   0   9   0
                            4   0   0   2   2  84   4   0   5   3  11
                            5   7   1   2   6   0  64   3   0   3   1
                            6   3   0   3   0   2   1  94   0   1   0
                            7   0   0   2   2   1   0   0  94   0  10
                            8   2   0   5   3   5   0   0   0  79   5
                            9   0   0   1   2  15   1   0  12   2  75
```

Accuracy = 0.827619

Discussion

The Multinomial Logistic Regression (MLR) model is basically a generalized version of the logistic regression model. That is, unlike the logistic regression model which tries to predict the outcome out of two possible choices, the multinomial logistic regression model tries to predict the outcome out of more than two possible choices. The MLR model starts out with a linear regression model that takes in numerical inputs, just like the logistic regression model, and this linear regression component allows for the MLR model to accommodate the columns with just 0s in the data set with numeric pixel values by just setting the parameters associated with those columns also to zero. However, the accuracy of the MLR model here is at best 0.7352381, or roughly 73.5%.

The Random Forest (RF) model first generates several preliminary decision tree (DT) models using subsets of the model-building data (with some observations possibility repeated). Those subsets then utilize a reduced number of dimensions. The RF model then aggregates the results of the DT models into one final RF model. This whole process is an attempt to avoid possible biases or overfitting with a DT model. It turns out that the accuracy of both numeric RF model and binary RF model are very similar at just under 0.95 or 95%. The binary RF model actually took less time to build than the numeric RF model. This is likely due to the fact that binary data is "flatter" with only 0 and 1 as possible pixel values while the numeric binary has a range of 0 to 255 for the pixel values. However, when factoring in the time required to transform the numeric data to the binary data, the numeric RF model takes far less time to produce.

The Naïve Bayes (NB) model classifies each observation using probabilities, where a particular label for an observation is likely to arise based off the probabilities associated with the pixel values of that observation. However, there is an assumption that the probability of one pixel having a particular value is independent of another pixel having a particular value, and realistically that is not the case. For instance, if a person writes the number 1, then several pixels down the center of that digit will likely be present (i.e. the pixel values are not equal to 0) because the writing tool covers a certain area. Moreover, if a person writes that same digit in such a way that one pixel down the center has a particular pixel darkness – let's say 150 – then the other pixels are likely to have a similar, if not identical, pixel darkness by virtue of how the digit was written. This lack of independence in the probabilities would result in the low accuracy level for the numeric NB model of 0.52, or 52%. However, the accuracy of the NB model increases to just under 0.83, or 0.83%, when using the binary data. This is because rather than a probability being calculated for one pixel being some integer value between 0 and 255 – let's say 100 – and a probability being calculated for another pixel for another integer value – let's say 200 – probabilities are being calculated for the same two pixels for either existing (i.e. the pixel value equals 0) or not (i.e. the pixel value equals 1).

Other Considerations

Support Vector Machine (SVM) models were attempted with both the numeric and the binary training data subsets; however, errors arose with both data types. The model-generating function requires some degree of variance in the values, so the columns with all 0s in the case of the numeric data prevented a numeric SVM model from being generated. In the case of the binary data, the model-generating function requires factors with more than one level for the sake of contrast, so the columns with 0s also prevented a binary SVM model from being generated.

Neural Network (NN) models were attempted as well with both the numeric and the binary training data subsets; however, errors once again arose with both data types. With respect to the numeric data, NaN ("Not a Number") values arose in the columns originally comprised of just 0s after all of the numeric valeues were scaled, a prerequisite for generating the NN model. Because any mathematical operation performed on an NaN value results in another NaN, calculations were expected to go awry when generating the numeric NN model, so the model-generation function was not run. With respect to the binary data, the model-generating function requires a vector or a matrix of numeric or complex values, not factors with levels. At any rate, past experience with generating NN models would lead to the expectation that times for generating models with either of the pixel data would be long, especially considering the number of observation and dimensions of the pixel data.

No attempt was made to generate Deep Learning (DL) models with either the numeric data or the binary data. Because Deep Learning is basically an extension of Neural Networks, the same issues that arose when attempting to generate the NN models were expected to arise again when attempting to generate the DL models.

With respect to the models that weren't generated as a function of the columns with just 0s in either the numeric or binary data sets, those columns with just 0s could just simply be removed prior to feeding the data sets into the model-generating functions. However, that approach seemed somewhat simplistic, since a column that might have had pixel with a 0 before could very well end up having a non-zero pixel later on as function of how someone writes out a digit. Therefore, all pixels were considered important for analysis.

Conclusion

Out of all of the classification models successfully generated, the model used for classifying the unlabeled data set ("train.csv") was the numeric Random Forest model (RF-Numeric) because of its high accuracy (nearly 95%) and its ability the process the unlabeled data set without any further transformation of pixels values.

For comparison, here is a table of the models, the input data type, and the accuracies of those models with that particular data type:

Classification Technique	**Input Data Type**	**Accuracy**
Multinomial Logistic Regression	Numeric	0.7352381
Multinomial Logistic Regression	Binary	N/A
Random Forest	Numeric	0.9495238
Random Forest	Binary	0.9457143
Naïve Bayes	Numeric	0.52
Naïve Bayes	Binary	0.827619

The predictions of the RF-Numeric model was saved to a separate .csv file ("Brian_Mallari_final_project_submission.csv") consisting of two columns: "ImageID" with the indexes of the unlabeled observations, and "Label" with the predicted label for a given observation."

Additional Remarks

Training machines to classify hand-written digits is a good starting point with respect to helping people better navigate the world around them. The data sets provided by MNIST (or the Modified National Institute of Standards and Technology) serve as a reliable benchmark for classification techniques, and as those classification techniques improve, they can be extended to other hand-written information, such text written in either native or foreign languages. From here, understanding and communication between culturally or linguistically different groups of people can improve, thus better facilitating relations in a variety of domains, such as tourism, diplomacy, and international commerce.

R Script

The .R file for this assignment ("Brian Mallari - 590 - Final Project.R") was submitted as an attachment along with a copy of this report.

Supplemental Files

The following .csv files were submitted as attachments along with a copy this report:

1. train.csv
2. test.csv
3. digit_train_subset_mod_pixel_off_or_on.csv
4. Brian_Mallari_final_project_submission.csv

Note: The first three files are necessary for the proper execution of the above .R file. The last file, however, will be generated each time the .R script has been run, but has been added nonetheless for easy access.

References

1.) Mathematics and R code were based off of notes taken for the following video podcasts by Dr. Chirag Shah:
    a. 3.1 Logistic Regression – lecture
    b. 4.3 Decision Trees and Random Forest – lecture
    c. 4.5 Random Forest – practice
    d. 5.1 Naïve Bayes – lecture
    e. 5.2 Naïve Bayes – practice
    f. 9.1 Support Vector Machine – lecture
    g. 9.2 SVM – practice
    h. 10.1 Neural Networks - lecture
    i. 10.2 Neural Networks – practice
2.) Additional theory and R code were based off of notes taken for the video podcast, 11.1 Deep Learning, by Ph.D. candidate Manasa Rath.
3.) Some R code was taken from the following assignment by Brian Mallari:
    a. Class 10 homework assignment on Neural Networks
    b. Class 11 homework assignment on Deep Learning
4.) Information on making a histogram from a data table in R was taken from the following webpage: https://stackoverflow.com/questions/36759077/make-histogram-from-data-table-in-r
5.) Information on the barplot() function in R was taken from the following webpage: https://www.statmethods.net/graphs/bar.html

6.) Information on counting unique values in a vector was taken from the following webpage: https://stackoverflow.com/questions/4215154/count-unique-values
7.) Information on control spacing between bars in a barplot was taken from the following webpage: https://stackoverflow.com/questions/13932708/how-do-i-control-space-between-bars
8.) Information on adding color to a histogram generated using the base plotting system in R was taken from the following webpage: http://www.r-tutor.com/elementary-statistics/quantitative-data/histogram
9.) Information on looping through the elements of a matrix was taken from the following webpage: https://campus.datacamp.com/courses/intermediate-r-for-finance/loops-3?ex=11
10.) Information on writing a .csv file using R was taken from the following webpage: https://stat.ethz.ch/R-manual/R-devel/library/utils/html/write.table.html
11.) Information regarding the 'sep' argument for the write.csv() function in R was taken from the following webpages:
    a. https://stackoverflow.com/questions/42946430/attempt-to-set-sep-dec-ignored-error-in-write-csv-format-r/42946579
    b. http://r.789695.n4.nabble.com/Help-with-writing-data-to-csv-td4629436.html
12.) Information regarding the extra column of row names when writing a .csv file was taken from the following webpage: https://stackoverflow.com/questions/7065648/write-table-extra-column
13.)
14.) Information on multinomial logistic regression was taken from the following webpage: https://en.wikipedia.org/wiki/Multinomial_logistic_regression
15.) Information on doing multinomial logistic regression in R was taken from the following webpage: http://r-statistics.co/Multinomial-Regression-With-R.html
16.) Information on an error relating to contrasts only working with factors with two or more levels when generating a multinomial logistic regression model with the nnet package in R was taken from the following webpage: https://stats.stackexchange.com/questions/141674/r-gives-me-the-error-contrasts-can-be-applied-only-to-factors-with-2-or-more-le
17.) Information on an error relating to too many weights when trying to make a multinomial regression model with the nnet package in R was taken from the following webpage: https://stackoverflow.com/questions/36303404/too-many-weights-in-multinomial-logistic-regression-and-the-code-is-running-for
18.) Information on an error relating to Random Forest and predictors with over 53 categories was taken from the following webpage: https://stats.stackexchange.com/questions/157331/random-forest-predictors-have-more-than-53-categories
19.) Information on the naiveBayes() function found in the e1071 package for R was taken from the following webpage: https://www.rdocumentation.org/packages/e1071/versions/1.6-8/topics/naiveBayes
20.) Information on an example of using SVM to classify data in R was taken from the following webpage: https://rischanlab.github.io/SVM.html
21.) Information on an error relating to length of center having to be the same as the number of columns in x when scaling numeric data for Neural Network modeling in R was taken from the

following webpage: https://stackoverflow.com/questions/36748003/scaling-data-in-r-resulting-in-error-length-of-center-must-equal-the-number?rq=1

22.) Information on an error relating to an int() argument having to be a string, a bytes-like object, or a number when scaling numeric data for Neural Network modeling in R was taken from the following webpage: https://stackoverflow.com/questions/39429579/typeerror-int-argument-must-be-a-string-a-bytes-like-object-or-a-number-not?rq=1

23.) Information on NaN in R was taken from the following webpage: https://www.r-bloggers.com/difference-between-na-and-nan-in-r/

24.) Information on an error regarding ‘row.names’ not being a character vector of a particular length was taken from the following webpage: https://stackoverflow.com/questions/48845309/row-names-is-not-a-character-vector-of-length