Assignment #3: Multidimensional Scaling and Self-Organizing Maps

Harini Anand

COMPONENT 1: Multidimensional Scaling

1. Introduction

For assignment 3, we first explored the multivariate technique called Multidimensional scaling (MDS). MDS is used to obtain a geometric or spatial model for the multivariate data by composing a similarity or proximity matrix of the data. For the first part of the assignment, we used the Recidivism data set, which is a collection of data of 1445 convicts and whether they committed a reoffence.

2. Data

The data set used for MDS analysis pertains to a random sample of convicts released from prison between July 1, 1977, and June 30, 1978. The primary purpose of the data collection was to obtain the time until they return to prison. The information was collected retrospectively in April 1984 using the records. The maximum length of observation is 81 months. There are 1445 observations in the data set with 18 variables. Table 01 lists the variables in the Recidivism data set.

Variable	Description
black	1 if black
alcohol	1 if alcohol problems
drugs	1 if drug history
supervised	1 if release supervised
married	1 if married when incarcerated
felony	1 if felony sentence
workprg	1 if N.C. prison work program
property	1 if property crime
person	1 if crime against person
nbr_piors	No. of prior convictions
education	Years of schooling
nbr_rules	No. of rules violations in prison
age	in months
time_served	time served, rounded to months
follow_up	length follow period, months
duration	max (time until return, follow)
censored	1 if duration right censored
log_duration	log (duration)

Table 01: Variables in Recidivism data set along with their description

In the data set, there are 10 nominal variables – black, alcohol, drugs, supervised, married, felony, workprg, property, person, and censored. The rest of the variables are continuous or quasi-continuous in nature - namely nbr_piors, education, nbr_rules, age, time_served, follow_up, duration, and log_duration. The duration variable captures the time until the convicts returned to prison. log_duration is obtained by taking the natural logarithm of the duration variable.

3. Data preparation

As part of the data preparation, we checked for the presence of any records in the data with missing values. The data set shows no such records. Then, we proceeded to add two new variables: ID and label. The ID variable contains a monotonically increasing integer to identify the row in the data set. The label variable is a concatenation of the ID, a collection of codes corresponding to the

nominal variables (black, alcohol, drugs, supervised, married, felony, workprg, property, person, and censored) that are true, the number of prior convictions, the number of years of education, age in months, time served in months, and duration in months. Table 02 shows the set of codes for the nominal variables.

Variable	Code added to the label if variable in the left column contains 1
black	bl
alcohol	al
drugs	dr
supervised	su
married	ma
felony	fe
workprg	wp
property	pr
person	pe
censored	cn

Table 02: Code added to label depending on variable containing a value of 1

For example, the 10th row in the data set with black variable value of 1, drugs variable value of 1, person variable value of 1, with 2 priors, 10 years of education, 253 months in age, 6 months in time served, and 0 duration will have ID variable set to 10 and label variable set to 10_bl_dr_pe_2_pri_10_ed_253_age_6_tsr_0_du. The label variable is created to help with the interpretation of the MDS plots.

Section A.1. in the Appendix has the R code for data preparation.

4. Exploratory data analysis

In this section, we present the results of the exploratory data analysis conducted on the recidivism data.

Basic EDA shows the following distribution of the nominal variables with values 0 and 1.

	Black	Alcohol	Drugs	Super	Married	Felon	Workprg	Property	Person	Censured
No (0)	744	1142	1096	442	1076	991	773	1077	1368	552
Yes (1)	701	303	349	1003	369	454	672	368	77	893

Table 03: Distribution table of the nominal variables

From Table 03, we can glean that there 43 more records about non-black convicts in the data. There are 839 convicts more with the alcohol variable set to 0 than the convicts with the variable set to 1. There are also about 749 more convicts with the drugs variable set to 0 than the convicts with the drugs variable set to 1. But when we compare the supervised release variable, there are 561 more convicts with supervised release. There are also more un-married convicts (707 more records) in the data. There are also fewer records of felons (537 fewer records) in the data. There are 101 fewer convicts in the N.C. work program. Only 368 records exist with property crime convictions. There are only 77 convicts that have committed a crime against a person. There are 341 more records of convicts who were censured. FIG 01 shows the histograms of these variables.

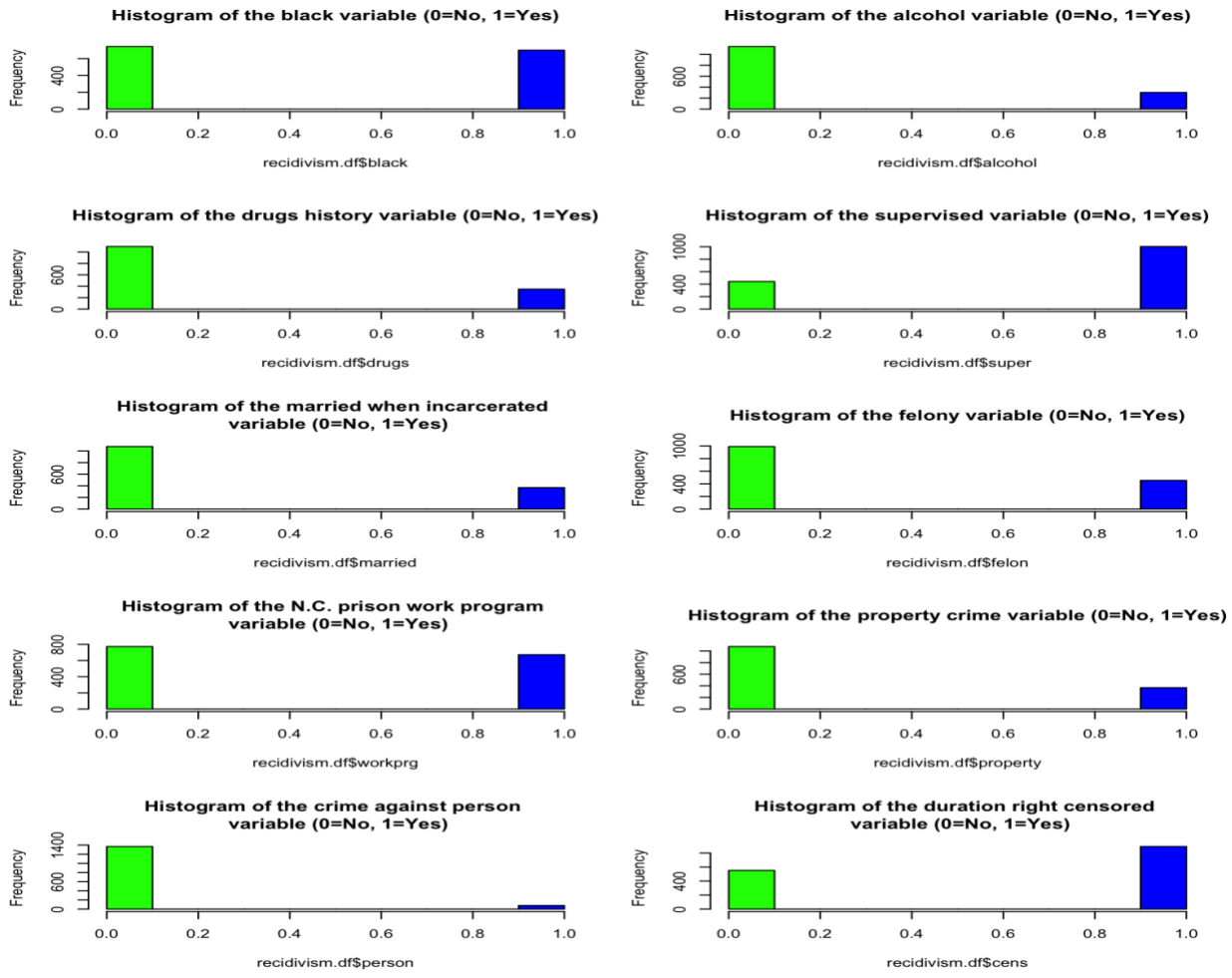

FIG 01: Histogram of the nominal variables - black, alcohol, drugs, supervised, married, felony, workprg, property, person, and censored.

Next, we ran the EDA on the number of prior convictions, numbers of years of schooling (education), number of rule violations in prison, and age in months. In the data, as shown in Table 04, there are 817 convicts with no prior convictions and 628 convicts with one or more prior convictions (the maximum being 28). There are 231 convicts with one prior conviction and 138 convicts with two prior convictions. From FIG 02 boxplot for the number of prior convictions, we can note data for the number of prior convictions is positively skewed with a large number of outliers on the upper end. The mode is 0.

Priors = 0	Priors > 0
817	628

Table 04: Distribution table for the prior convictions variable

The EDA on the education variable shows the range for the number of years of educations ranges from 1 year to 19 years. The mode is 12, with 280 convicts. The median is 10 years of education. The distribution has some negative symmetry with long tails on the left. For the

education variable too, there are a large number of outliers on either side of the distribution. Table 05 shows the distribution table for the education variable.

1 year	2 years	3 years	4 years	5 years	6 years	7 years	8 years	9 years	10 years	11 years	12 years	13 years	14 years	15 years	16 years	17 years	18 years	19 years
3	12	20	26	20	44	77	180	244	265	193	280	26	30	11	9	1	3	1

Table 05: Distribution table for the education variable

Next, we performed basic EDA on the rules variable, which captures the number of rule violations committed by the convicts in person. From Table 06, we can gather there are 817 convicts with no rule violations. There are 628 convicts with one or more rules violations. There are 294 convicts with 1 rule violation and 133 convicts with 2 rule violations. Similar to the prior convictions variable, the rule violations variable is positively skewed with a large number of outliers on the upper end of the distribution. The mode is at 0. FIG 02 shows the histogram and the boxplot for the rules variable.

rules = 0	rules > 0
806	639

Table 06: Distribution table for rules variable

The age variable of the convicts ranges from 198 months (=16.5 years) to 933 months (=77.75 years). The mode for the age variable is 253 months (=21.08 years). The mean is 345.4 months (=28.78 years), and the median is 307 months (=25.58 years). FIG 02 shows the age data is slightly positively skewed with long tails. It also has a large number of outliers on the right side.

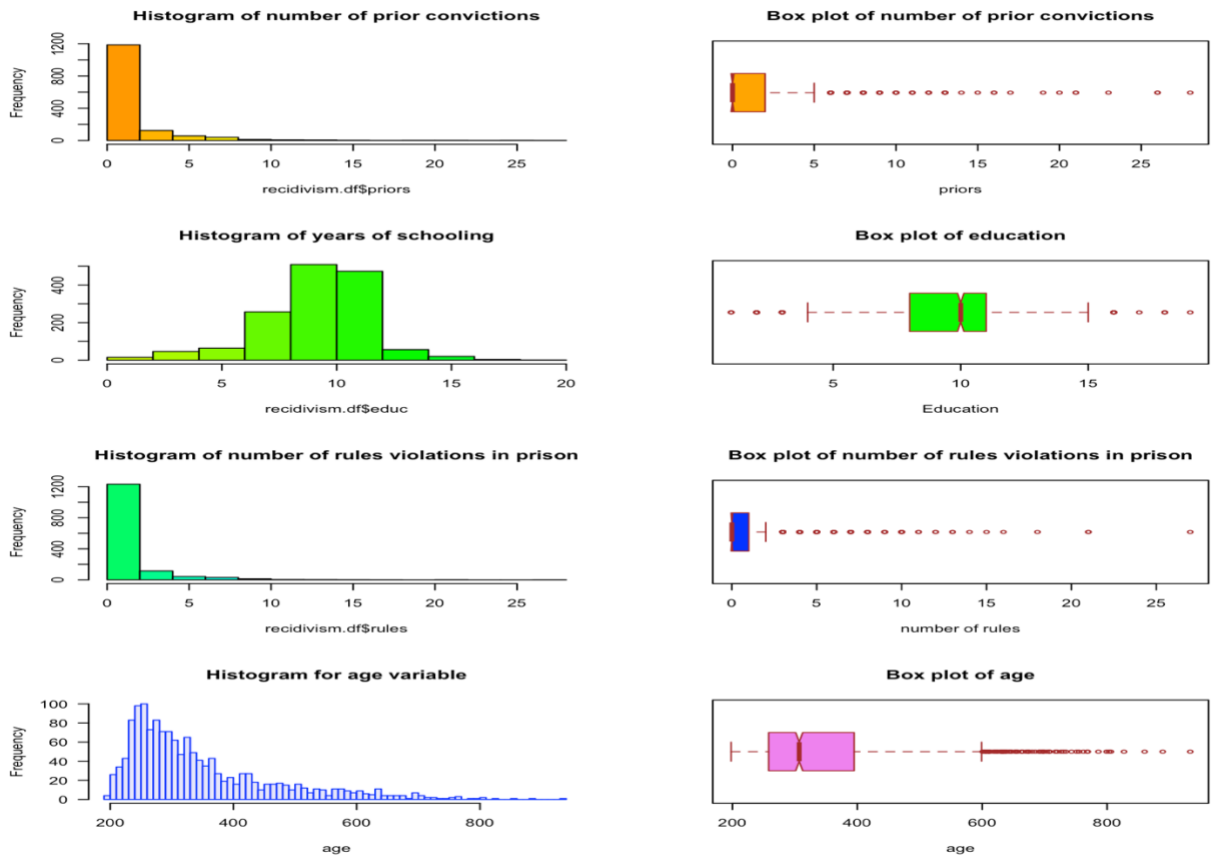

FIG 02: Histogram and boxplot of the variables -priors, education, rules, and age.

Next, we looked at the variables – time served, the number of months followed, duration in months until the convict returned to prison, and the logarithm of duration.

The EDA for time served shows it ranges from 0 to 219 months (= 18.25 years). The mode for time served is 6 months. The median is 12 months. The mean is 19.18 months. The histogram and the box plots for the time served variable in FIG 03 shows positive skewness with long tails and a large number of outliers on the higher end.

The follow variable shows the convicts were followed for a maximum of 81 months. The minimum is 70 months. The mode is 70 months. FIG 03 shows slight positive skewness. However, there are no outliers. The median is 74 months. The mean is 74.89 months.

The durat variable ranges from 1 month to 81 months (the maximum time for which the convicts were followed). The mode is 73 months. The median is 71 months. The mean is 55.37 months. Unlike all the previous variables, the durat variable shows significant negative skewness with long tails to the left (on the lower end). This can be seen from the histogram and the boxplot in FIG 03. The durat variable has no outliers.

The last variable we looked at in the original set is ldurat (the logarithm of the durat variable). The values range from 0 to 4.394449. The mode is shown to be 4.29046. The mean is 3.745 and the median is 4.263. Like the durat histogram and boxplot, FIG03 shows significant negative skewness with long tails to the left. However, there are some outliers to the left of the distribution.

Section A.2. in the Appendix has the R code for exploratory data analysis.

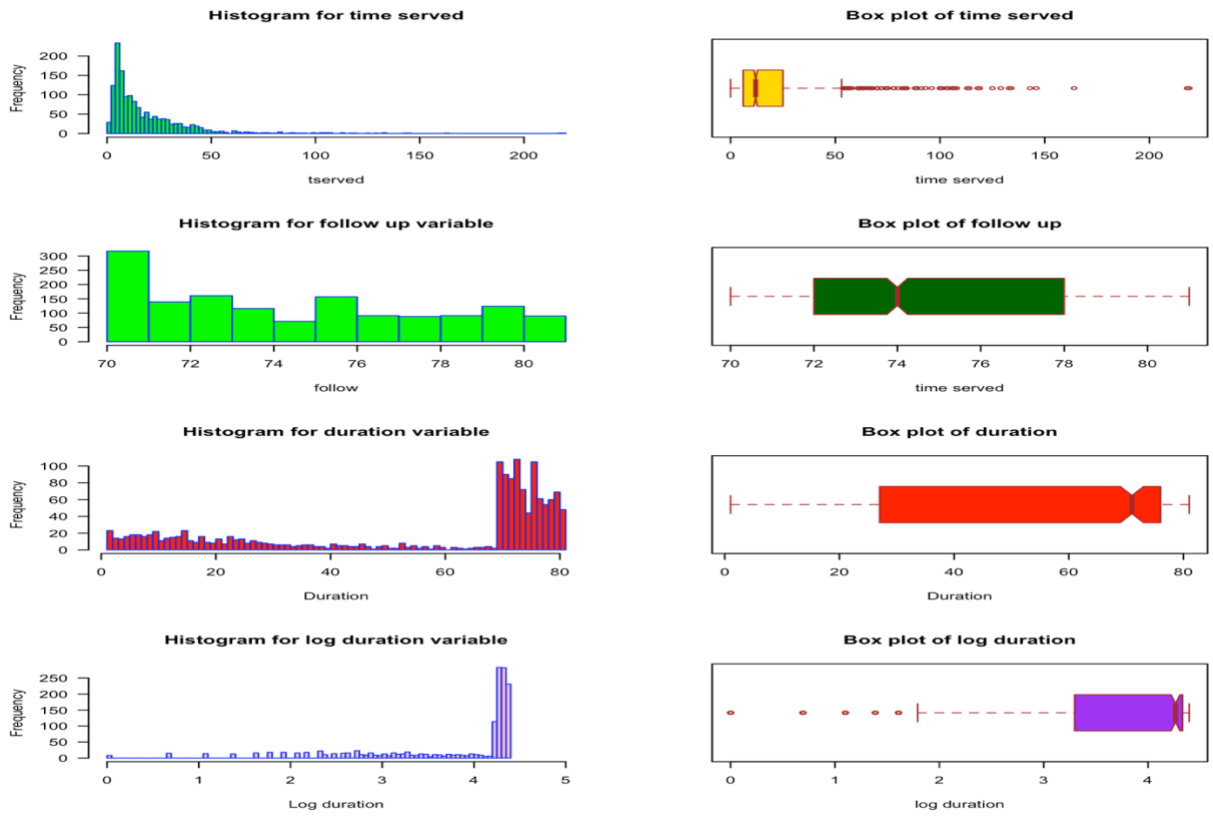

FIG 03: Histogram and boxplot of the variables -priors, education, rules, and age.

5. Dissimilarity Matrix using Euclidean distances

We next proceeded to compute the dissimilarity matrix using Euclidean distances. Before that, we chose to remove the following variables in the data set:

ID
black
alcohol
drugs
super
married
felon
workprg
property
person
cens
follow
ldurat
label

The reason to remove the nominal variables was to include only the continuous or quasi-continuous variables in the Euclidean distance computation. We also removed the ldurat variable because it just contains a log transformation of the durat variable.

The reduced data set has only the following variables:

priors
educ
rules
age
time served
durat

We used the R distance matrix computation function dist() with the method "Euclidean" to obtain the dissimilarity matrix. Upon converting the data to a matrix, we attempted to identify any patterns in that matrix. For that, we used a heatmap() R function. We used the heatmap to visualize the concentration of values in the dissimilarity matrix and to identify patterns in the data. The colors in the heatmap relate to the value. We have created two heatmaps – one without reordering of the rows or columns, and one with rearrangement of rows and columns so that similar values are clustered.

FIG 04 presents the heatmap of the dissimilarity matrix with no row/column arrangement. In this figure, we noted that some records in the data have higher distance values, and these are represented using brighter red colors on the map. Interestingly, the higher distance values form bright color lines on the map.

FIG 05 presents the heatmap of the dissimilarity matrix with a hierarchical clustering dendrogram on the side. In this figure, the rows and columns are sorted. Row dendrograms show the similarity between rows and the clusters computed using hierarchical clustering. Column dendrograms show the similarity between the columns and clusters computed on the columns using the hierarchical clusters. A point to note is, in the case of classical multidimensional scaling, the distance (proximity) matrix is symmetric.

Section A.3. in the Appendix has the R code for the computation of the dissimilarity matrix.

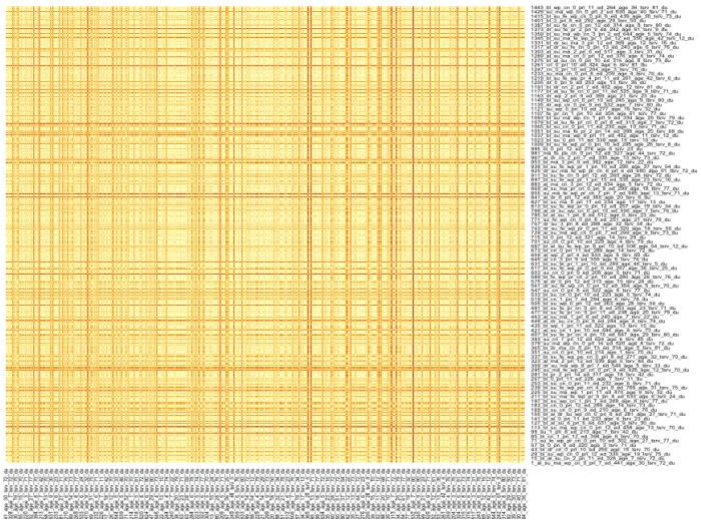

FIG 04: Heatmap of the dissimilarity matrix created using Euclidean distance with no row/column reordering

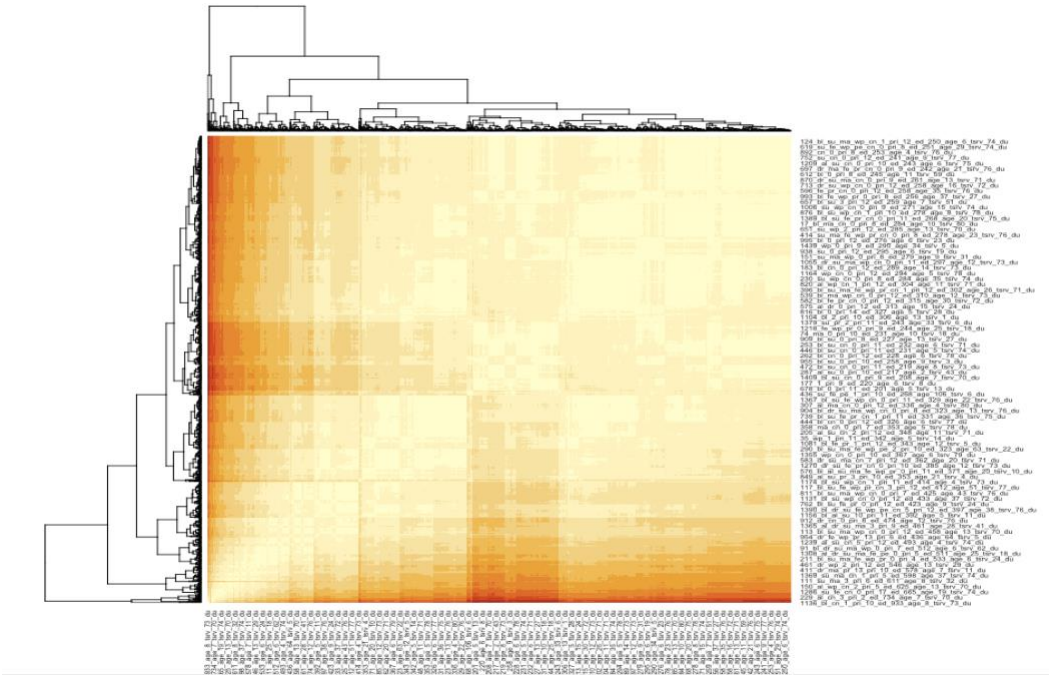

FIG 05: Heatmap of the dissimilarity matrix created using Euclidean distance with row/column rearrangement and with hierarchical clustering applied

6. Classical multidimensional scaling using the Euclidean Distances dissimilarity matrix

Using the computed Euclidean distance matrix, we proceeded to compute the classical multidimensional scaling model. This was done using the cmdscale() R function. We set k=2 for the number of dimensions and eig=TRUE for eigen values. The classical MDS, also known as the Principal Coordinates Analysis, takes the Euclidean distance matrix as input and outputs the geometric configuration by minimizes a loss function. **The GOF (goodness-of-fit) value returned by cmdscale()**

for the model is 0.9737015, which shows a good fit. However, there are negative eigenvalues put out by cmdscale() for this model. FIG 06 shows the eigenvalue plot. It shows how much variance is captured by each dimension. From FIG 06, we can see Dimension 1 explains most of the variance.

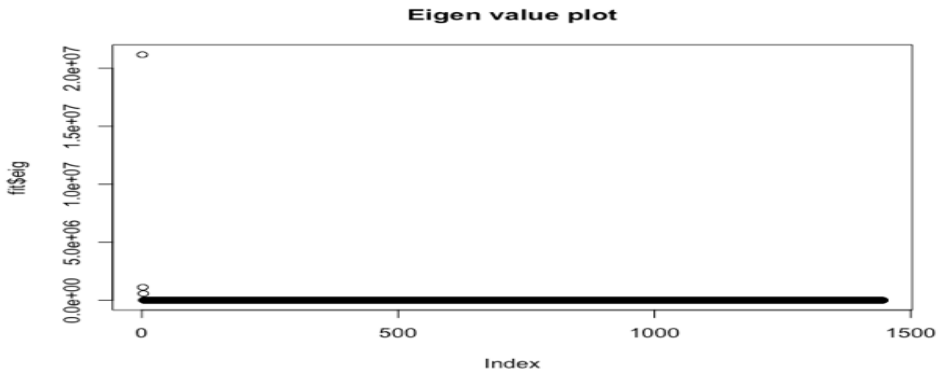

FIG 06: Eigen value plot for cmdscale model

The geometric configuration for the dissimilarity matrix is shown in two plots below. FIG 07 shows the geometric configuration with no labels. FIG 08 is drawn for the same geometric configuration with the labels so that we could provide an interpretation.

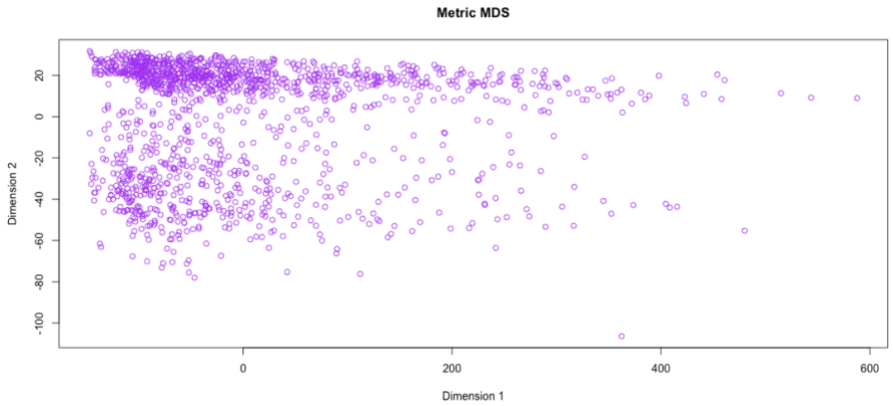

FIG 07: Classical MDS solution with no labels

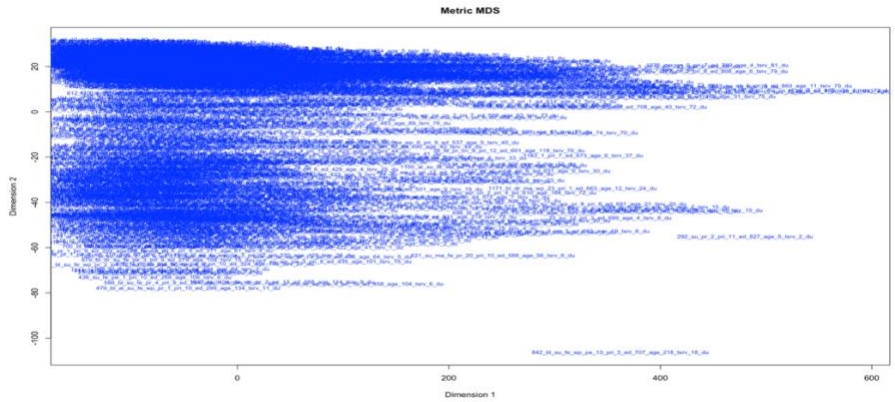

FIG 08: Classical MDS solution with added labels to the data points

Interpretation:

In the classical MDS solution above, there are two areas where we noted a concentration of the geometric points. Both these areas are to the left in the plot with the biggest concentration to the top, above the Dimension 2 = 0 line. The second smaller concentration is below the Dimension 2 = 0 line (close to Dimension 2 = -40) and to the left of Dimension 1 = 0 line.

Using the added labels in FIG 07, we derived some interpretation based on the patterns of these points.

Generally,

Data points of convicts with "black" set to 1: Decrease going from left to right along Dimension 1

Data points of convicts with "property crime" set to 1: Decrease going from left to right along Dimension 1

Data points of convicts with "person crime" set to 1: Decrease going from bottom to top along Dimension 2

Data points of convicts with "censured" set to 1: are only noted above the line Dimension 2 = 0 (top half of the plot)

Age of the convicts is noted to increase from left to right along Dimension 1

Time served of the convicts is observed to decrease from left to right along Dimension 1

Duration before the convicts returned to prison increases from bottom to top along Dimension 2.

No discernible patterns were noted with priors, education, felony, work program, supervised, alcohol, drugs, or married.

Base on this, we conclude the Dimensions 1 and 2 carry a compound meaning involving age, time served, race, type of crime, etc.

Section A.4. in the Appendix has the R code for classical MDS.

7. Non-metric multidimensional scaling

In this section, we present the results of the non-metric multidimensional scaling obtained with the reduced data set with the variables - priors, educ, rules, age, time served, and durat. We used the ***isoMDS()*** R function with the Euclidean distance dissimilarity matrix computed in section 5. We used k=2 for the number of dimensions. Non-metric MDS assumes a rank order in the proximities. FIG 09 shows the plot of the rank order of the proximities to the actual proximities.

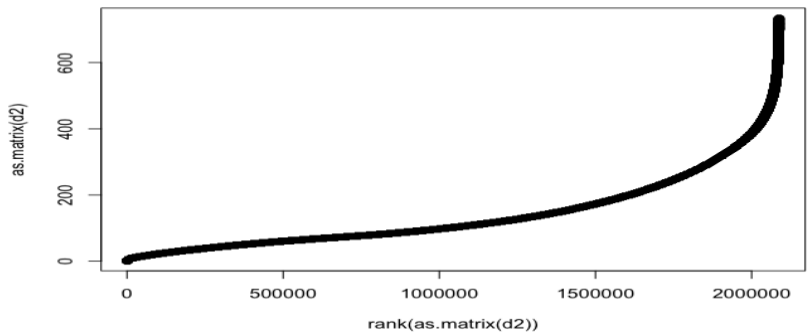

FIG 09: plot of the rank order of the distance to the actual distance

The initial stress value for two-dimensions was 4.670458, and the stress value returned upon convergence for non-metric MDS model is 4.67033.

The plots of the model produced geometric configuration for the dissimilarity matrix is shown below. FIG 10 shows the same plot (spatial mapping) with no labels. FIG 11 is plotted with the labels added so that we could provide an interpretation.

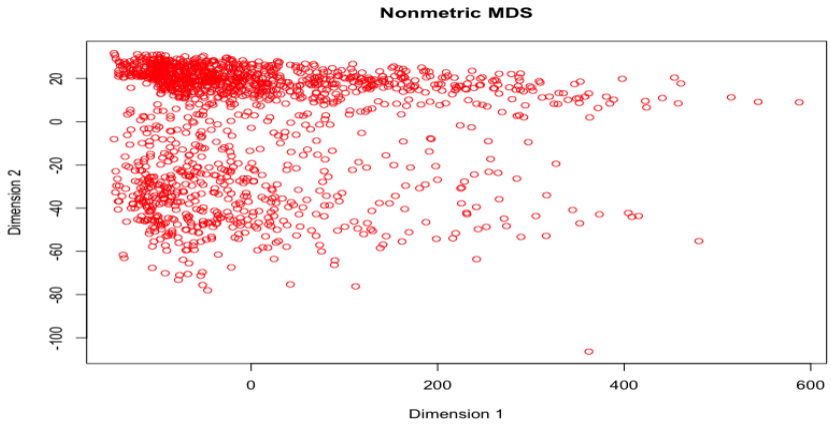

FIG 10: Non-metric MDS solution with no labels

Interpretation and comparison with the classical approach:

An important point to note is that the geometric configuration generated by non-metric MDS is exactly the same as that of the classical MDS. So, the spatial mapping obtained using metric distance (in the case of classical MDS) and rank distance (in the case of non-metric MDS) is the same. Therefore, the interpretation provided in the previous section applies to these mappings.

Section A.5. in the Appendix has the R code for non-metric MDS.

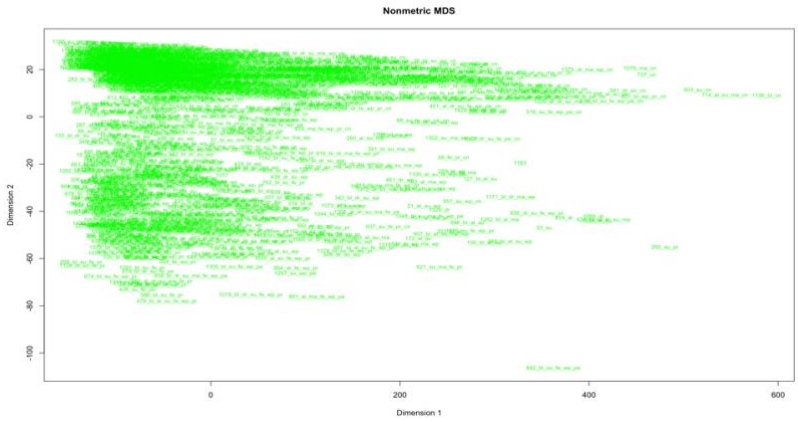

FIG 11: Non-metric MDS solution with added labels to the data points

8. Ramsay Multidimensional Scaling

In this section, we present the results of an MDS model created using Ramsay's method. Ramsay proposed using likelihood methods in multidimensional scaling and the calculation of confidence regions in the solution. An assumption made with this technique is that true dissimilarities

correspond to weighted Euclidean distances for a set of weights. For this exercise, we used the monotonic spline function transformation. To create the model, we used the mds() function in the R SMACOF package. We passed in the Euclidean distance dissimilarity matrix to the function as a parameter. The number of dimensions, ndim, is 2. The output of the mds() function is shown in Table 07.

```
Call:
mds(delta = d1, ndim = 2, type = "mspline")

Model: Symmetric SMACOF
Number of objects: 1445
Stress-1 value: 0.03
Number of iterations: 27
```

Table 07: output of mds() function with mspline transformation

FIG 12 and FIG 13 show the spatial mapping produced by the above model. Fig 12 shows the geometric points with no labels. Fig 13 has the labels added.

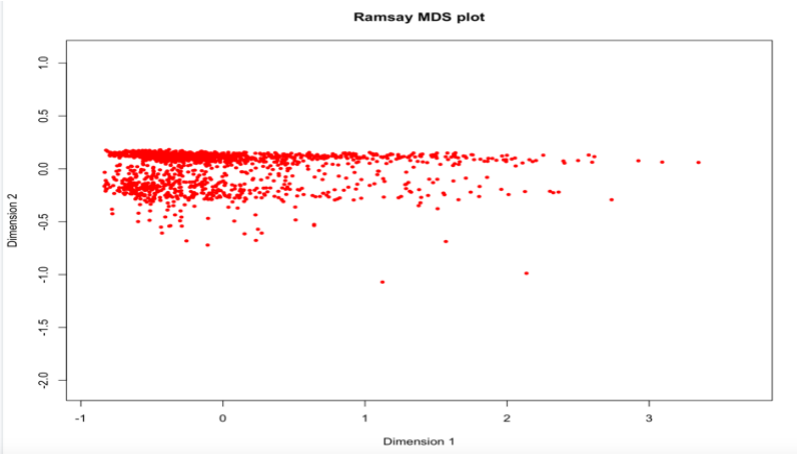

FIG 12: Ramsay MDS solution with no labels to the data points

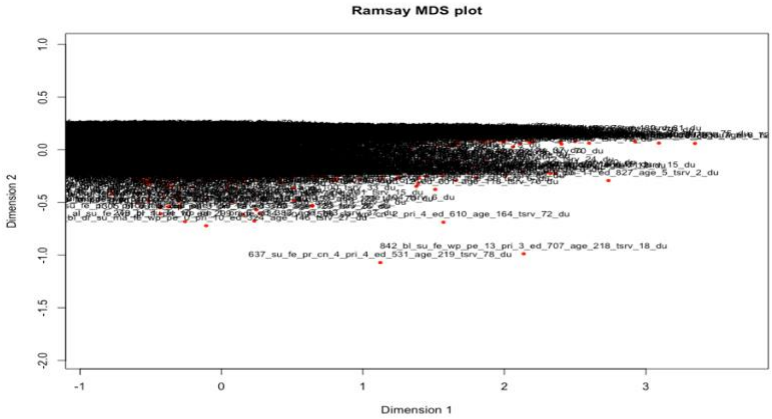

FIG 13: Ramsay MDS solution with labels added to the data points

We also attempted to create confidence regions (one of the suggestions proposed by Ramsay) in the resulted spatial mapping using the confEllipse() R method in SMACOF package. **However, with 1445**

observations, the confEllipse() function does not complete even after several hours. As a result, the confidence regions were not shown in the above plots.

Interpretation and comparison with the classical approach:

In FIG 12, which was created with mspline transformation, one of the transformations proposed by Ramsay, there are also two areas of concentrated points similar to what was noted with classical MDS and Non-metric MDS plots. However, Dimension 1 and Dimension 2 have different scales than the classical MDS plot. The plot looks like a compressed version of the classical MDS plot. Due to the concentrated point mapping, it was difficult to obtain an interpretation of the dimensions.

Section A.6. in the Appendix has the R code for Ramsay MDS.

COMPONENT 2: Self Organizing Maps

1. Introduction

For the second part of assignment 3, we explored the multivariate technique called Self-Organizing Maps (SOM). Our objective of using SOM was to reduce the high-dimensional data to a lower-dimensional nonlinear manifold. SOM maps the projected data to a discrete set of interconnected nodes. Each node represents a grouping or cluster of relatively homogeneous points. For this assignment, we used the College Acceptance data set, which is a collection of data of 400 students' acceptance into various engineering programs.

2. Data

The data set used for SOM analysis contains college acceptance data of 400 students into various engineering programs. The data set has four variables. Table 01 shows the variables along with their descriptions.

3.

Variable	Description
admit	0 = Not admitted, 1 = Admitted
gre	Student's GRE score
gpa	Student's GPA
rank	College ranking

Table 01: Variables in College Acceptance data set along with their description

In the data set, there is one nominal variable – admit, with two values (0 and 1). The data set has one ordinal variable, rank, which indicates the college ranking. The other two variables, gre, and gpa are ratio variables.

3. Data preparation

As part of the data preparation, we checked for the presence of any records in the data with missing values. The data set shows no such records.

Section B.1. in the Appendix has the R code for data preparation and the helper functions.

4. Exploratory Data Analysis (EDA)

In this section, we present the results of the exploratory data analysis conducted on the college acceptance data.

Basic EDA for the admit variable shows the following distribution. From Table 02, we can glean there is more data (about 36.5% more) for students who are not admitted than those who are admitted.

Admit value = 0	Admit value =1
273	127

Table 02: Distribution table for the admit variable

Next, we looked at the gre variable. gre has a median of 580 and a mean of 587.7. The histogram and boxplot for gre in FIG 01 show a slight negative skewness. The data is unimodal. In addition to that, the data has a few outliers on the lower side.

Following that, we looked at the gpa variable. gpa has a median of 3.395 and a mean of 3.390. The histogram and boxplot for gpa also show some negative skewness in FIG 01. gpa data is also unimodal. The data has a few outliers too on the lower side.

Lastly, we performed EDA on the college ranking variable. The rank variable takes four values: 1, 2,3,4. The mode is 2. The data has mild positive skewness. FIG 01 has the histogram and boxplot for rank data.

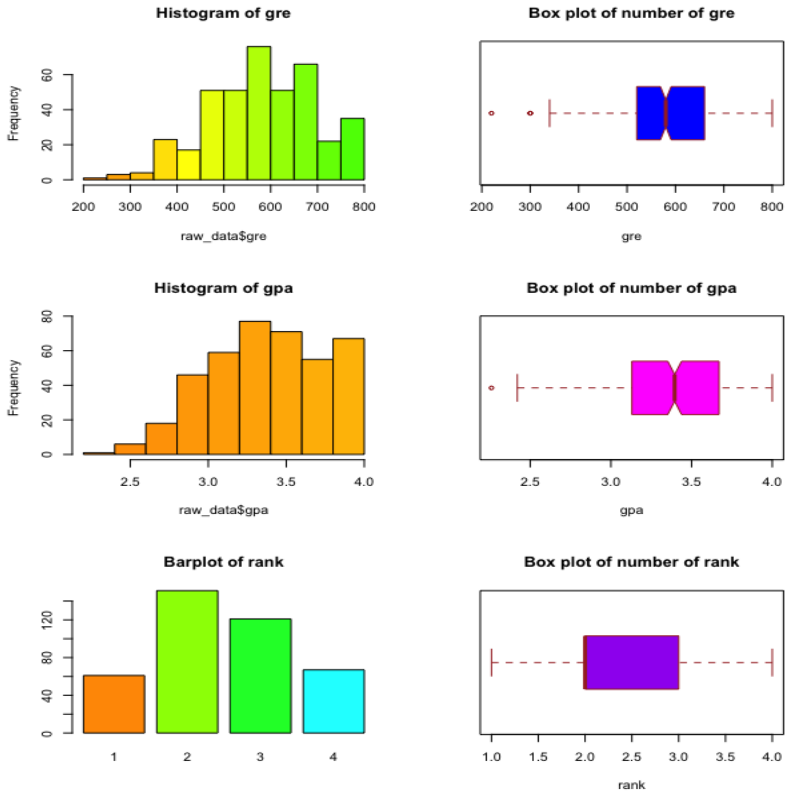

FIG 01: Histogram and boxplots for the variables – gre, gpa, and rank

FIG 02 below shows the density ridge plot based on values of admit variable values - “not admitted” and “admitted” densities corresponding to the gre, gpa, and rank variables.

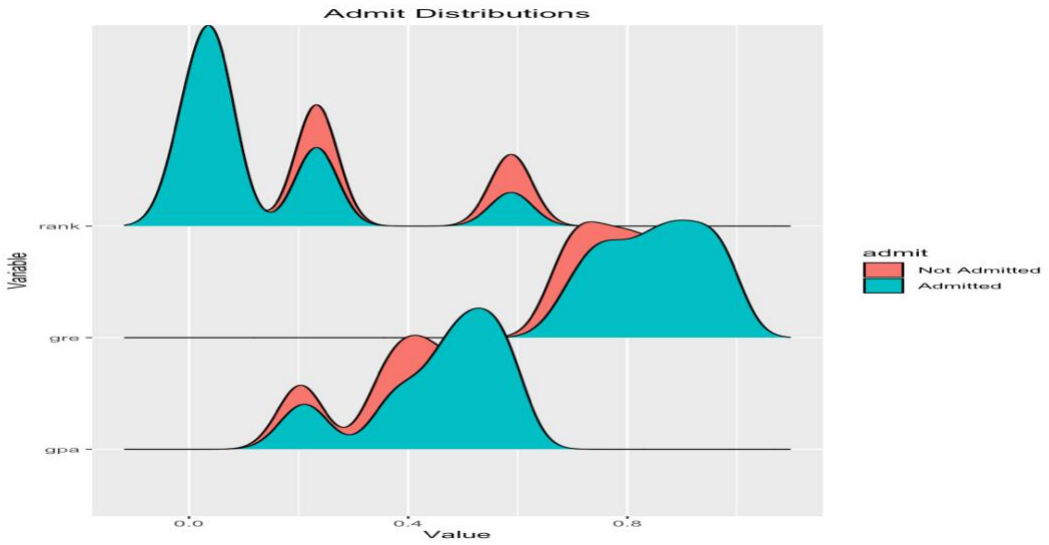

FIG 02: Distribution of not admitted and admitted student per variable

Additionally, we also perform normalization on the data. This is to ensure no one variable has an undue influence on the results just because it has a large variance or absolute value. We use the scale() R function to center and scale the numeric variables, namely gre and gpa variables in the data. We treat the rank variable as a factor variable.

For the SOM model, we will use the scaled continuous variables (gre and gpa) and the factor variable (rank) in the original data set. We excluded the admit variable because it is a response variable.

Section B.2. in the Appendix has the R code for exploratory data analysis.

5. Fitting a SOM Model

To fit a SOM model, we use the following parameter values:

Epoch: 2000

Grid size: 10x10 (computed using the number of sample size based on the formula 5 * sqrt(N) N=400 in our case)

Alpha (learning rate) = two values, 0.1 and 0.01

The data set has both numeric and factor variables. The numeric data uses the ***sumofsquares*** distance. The factor variable uses the ***tanimoto*** distance. Using the ***supersom()*** R function, we trained a SOM model.

Section B.3. in the Appendix has the R code for fitting a SOM model – 10x10 grid size.

6. Evaluation of the SOM Model

In this section, we evaluate the SOM model trained in section 5. We first evaluated whether the chosen epoch value of 2000 is sufficient. Based on the training progress or changes plot in FIG 03, we were able to determine the SOM model achieved the convergence (a minimum plateau). The mean distance to the closest unit decreases with iterations. By about 1700 iterations, it is close to 0. FIG 03 shows the training process of the model over the epoch we selected. Therefore, an epoch of 2000 is adequate for the model.

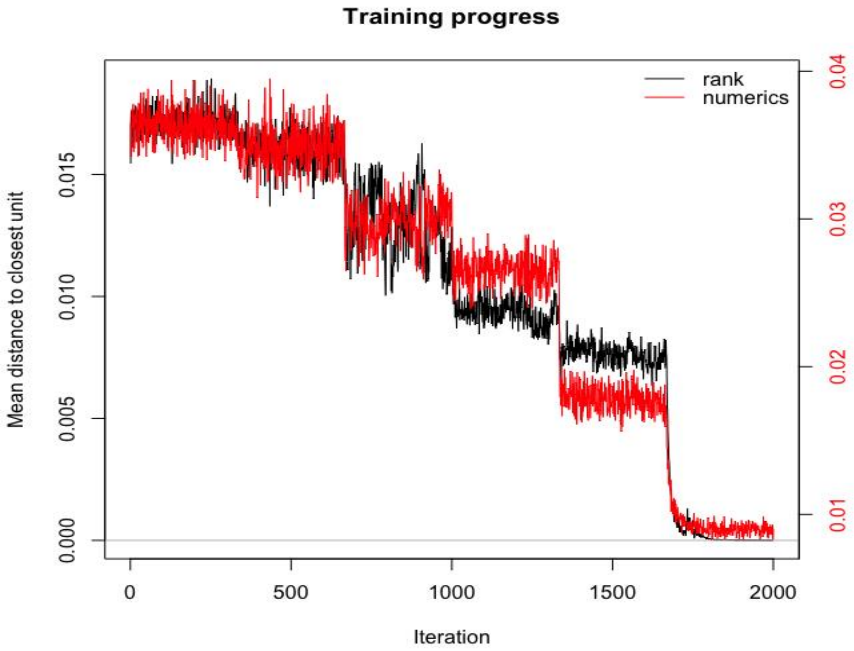

FIG 03: Training Progress for the SOM model (changes plot)

Next, we evaluated if the chosen grid size of 100 (10x10) is adequate. For this, we used the counts map, which provides a measure of the map quality. The counts map for the generated SOM model is shown in FIG 04. **Empty nodes indicate the chosen grid size is too big. Some gray or empty nodes are tolerated. However, large points per node is an indication that a larger grid size is needed. The goal is to not have more than 10 points per node.**

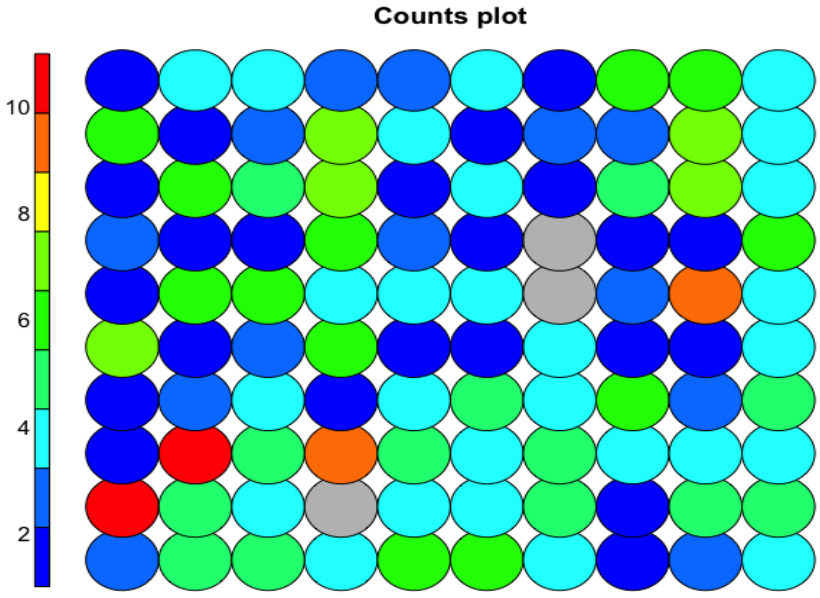

FIG 04: Counts plot for the SOM model with grid size 100 (10x10 grid)

In FIG 04, the nodes that need further evaluation are the gray color, orange color, and red color nodes. **The scale to the left of the plot shows the counts corresponding to the colors.** Using the ***get_node_counts()*** R package, we obtained the number of points assigned to these nodes by the model. Table 03 shows the counts for the nodes of interest.

Node	Color	Count of data points assigned to the node
11	Red	11
14	Gray	0
22	Red	11
24	Orange	9
57	Gray	0
59	Orange	9
67	Gray	0

Table 03: Counts per the red, gray, and orange nodes in the counts plot

The gray nodes are empty. They have 0 data points assigned. The red nodes have 11 data points assigned to the nodes. The orange nodes have 9 data points. Per Table 03, the gray nodes (empty nodes) indicate the grid size is too big, while the red nodes indicate a larger grid size may be needed. A small number of gray nodes are tolerated. **However, the red nodes have more than 10 points. Due to this, we consider the 10x10 grid size is not adequate.**

The average number of observations assigned to the nodes is 4.123711 (~ 4). The goal is to have not more than 10 points per node.

As a next step, we looked at the neighbor distance plot (U-Matrix). The neighbor distance plot helps to visualize the high dimensional data as a 2D image. FIG 05 shows the neighbor distance plot computed for the SOM model trained in section 5. In the plot, nodes having the lowest neighbor distance are most similar and have the same color. On the other hand, nodes having larger distances from their neighbors are dissimilar. The scale on the left of the plot shows the distance scale corresponding to the color. We looked for nodes that have larger distances from their neighbors or are dissimilar. We evaluated red and orange nodes as anomalies. Red nodes have a neighbor distance of 0.8 or more, and the yellow nodes have a neighbor distance of 0.7 or more.

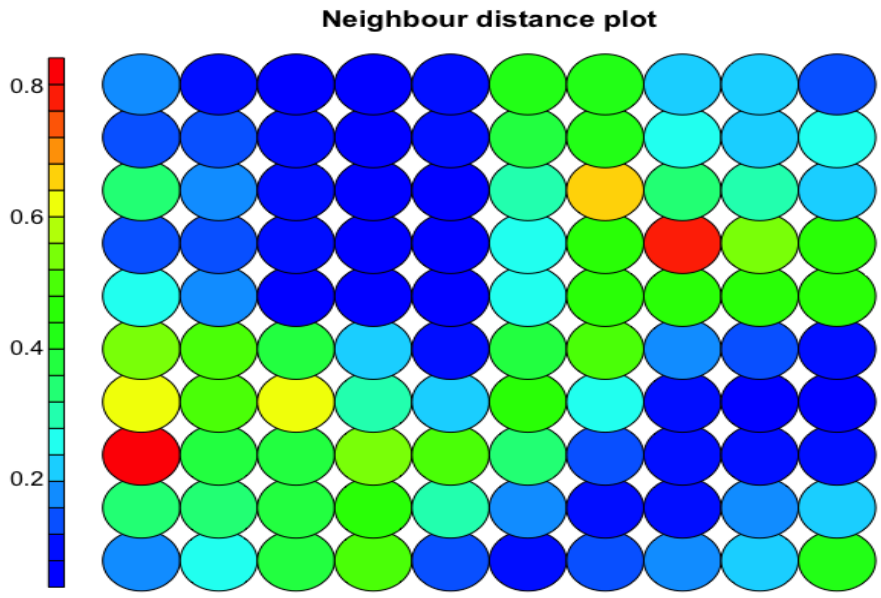

FIG 05: Neighbor distance plot (U-Matrix) obtained using the SOM model

Table 04 shows the anomalies associated with the three nodes (2 red, 1 orange). Further analysis is needed to determine why these nodes have higher neighbor distance.

Node	Color	Anomalies # admit gre gpa rank
(1,3)	Red	49 Not Admitted 440 2.48 4 290 Not Admitted 420 2.26 4
(7,8)	Orange	356 Admitted 760 2.81 1
(8,7)	Red	373 Admitted 680 2.42 1

Table 04: Anomalies per the red and orange nodes in the neighbor distance plot

Next, we explore the codes map to visualize the node weight vectors (codebook). FIG 06 shows the codes map obtained using the SOM model trained in section 5. The node weight vectors or codebooks are made up of normalized values of the original variables used to generate the SOM. Each node's weight vector is representative of the data points that are mapped to a node. FIG 06 shows two fan diagrams (one for rank and a second one for numerics), which depicts the patterns in the distribution of variables in the data. Individual fans are representations of the magnitude of each variable in the weight vector and is shown for each node. The fan size in each map is based on the values in the *codes* attribute of the SOM object.

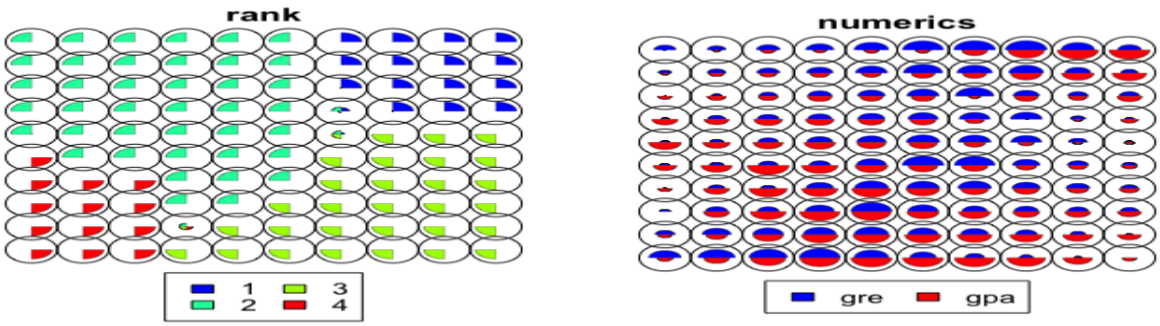

FIG 06: codes map obtained using the SOM model

In FIG 06, the map on the left shows how the college rank variable, which has four values (1,2,3, and 4) contributes to the codes or the node weights. We can note that the rank variable contributes a maximum of 25% to the node weights. In the majority of nodes, only one value of rank contributed to the node weight. However, there are three nodes in the rank map which deviate from this pattern. Rank value 2 has the most nodes because Rank value 2 is the most occurring value in the data. The map to the right shows the magnitude of the gre and gpa variables to the node weights. We can note that the maximum contribution for each of these variables (gre or gap) to the node weights is 50%. In the map to the right, there are also nodes where numerics have very little contribution to the node weights.

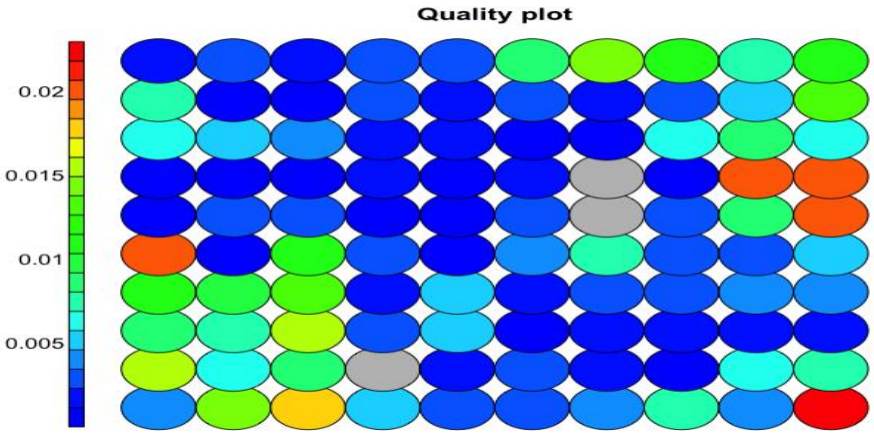

FIG 07: Quality plot for the SOM model

Lastly, we explored the quality plot (FIG 07) for the SOM model trained in section 5, which shows the mean distance of objects mapped to a unit to the codebook vector of that unit. The smaller the distances, the better the objects are represented by the codebook vectors. We note red and orange nodes in the Quality plot with mean distances in the range of 0.02.

Section B.4. in the Appendix has the R code for evaluating the SOM model.

7. Experiment results with SOM Model

In this section, we present the results of experimentation with grid size. In the previous section, we noted that the grid size of 10x10 had shown both empty nodes and nodes with more than 10 data points. The goal is to have not more than 10 data points per node. Based on that, we concluded that the grid size 10x10 is not adequate.

Due to the presence of red color nodes, indicating more data points per node than what we would desire, we chose to try a slightly bigger grid size. We chose to increase the grid size to 11x11 and re-train a SOM model. With the same Epoch value of 2000, the new SOM model with a map dimension of 11 reached convergence. A minimum plateau is noted around 1700 iteration in FIG 08.

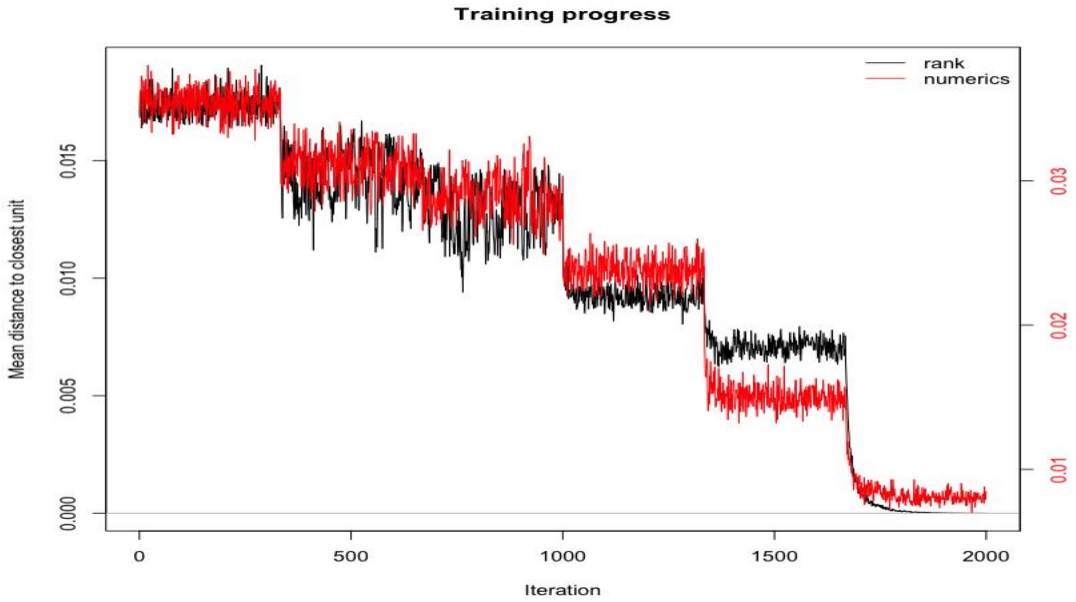

FIG 08: Training Progress of the re-trained SOM model with 11x11 grid size

FIG 09 shows the counts plot for the SOM model with an 11x11 grid size. In FIG 09, the nodes of interest are the gray color, orange color, and red color nodes. The scale to the left of the plot the scale of the counts corresponding to colors. Using the ***get_node_counts()*** R package, we obtained the number of points assigned to these nodes by the model. These counts are shown in Table 05.

Node	Color	Count of data points assigned to the node
4	Gray	0
42	Red	9
49	Orange	7
54	Orange	7
82	Gray	0
86	Orange	7
91	Orange	7

Table 05: Counts per the red, gray, and orange nodes in the counts plot of the new SOM model with 11x11 grid size

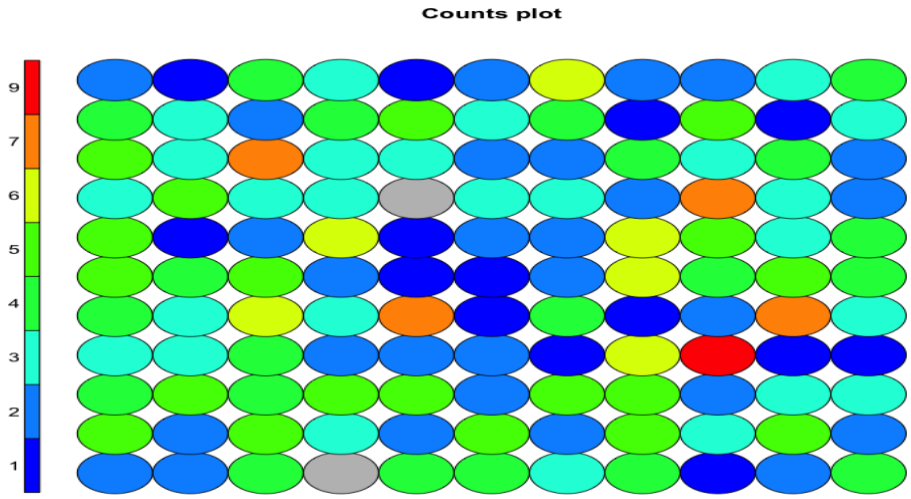

FIG 09: Counts plot for the SOM model with grid size 121 (11x11 grid)

From both Table 05 and the plot in FIG 09, we can observe that the red color nodes have only 9 data points assigned. **With the 11x11 grid, it meets the goal to have not more than 10 data points per node. With an increase in grid size, there are no concerns of overloading data points to a node.** The average number of data points per node is 3.361345.

Next, we looked at the neighbor distance plot (U-Matrix). FIG 10 shows the neighbor distance plot computed for the new SOM. This plot helps to visualize the data in a high-dimensional space using a two-dimensional display. In the plot, nodes having the lowest neighbor distance are most similar and have the same color. On the other hand, nodes having larger distances from their neighbors are dissimilar. The scale on the left of the plot shows the distance scale corresponding to the color.

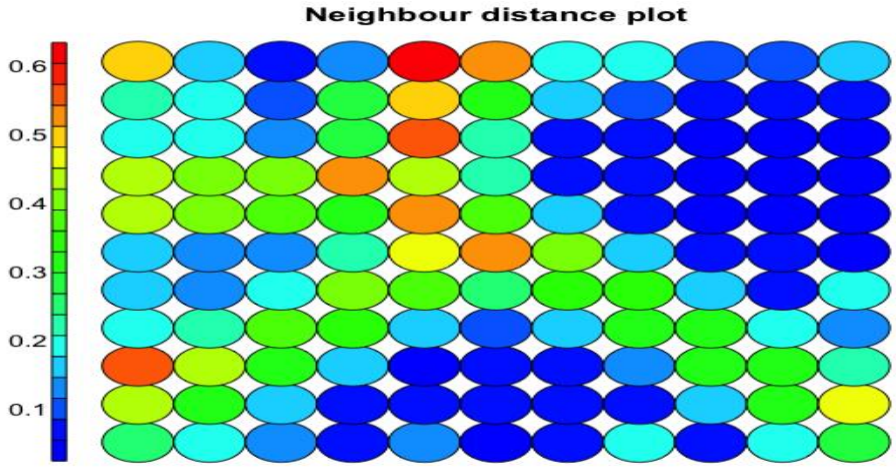

FIG 10: Neighbor distance plot (U-Matrix) obtained using the new SOM model

The scale on the left shows slightly smaller neighbor distance values for red and orange nodes for the new SOM model compared to the previous SOM model. From FIG 10, we noticed there is one red node, 2 bright orange nodes, and 3 light orange nodes whose neighbor distances are greater than 0.5. We explore the red and 2 bright orange nodes as anomalies.

Table 06 shows the anomalies associated with the three nodes (1 red, 2 bright orange). Further analysis is needed to determine why these nodes have higher neighbor distance.

Node	Color	Anomalies # admit gre gpa rank
(5,11)	Red	295 0 480 2.55 1
(5,9)	Bright Orange	152 0 400 3.38 2 233 0 380 3.38 2 349 0 400 3.36 2
(1,3)	Bright Orange	49 0 440 2.48 4 72 0 300 2.92 4 84 0 380 2.91 4 290 0 420 2.26 4

Table 06: Anomalies per the red and bright orange nodes in the neighbor distance plot

Next, we explore the codes map with the new SOM model to visualize the node weight vectors (codebook). FIG 11 shows the codes map obtained using the new SOM model. As mentioned before, the node weight vectors or codebooks are made up of normalized values of the original variables used to generate the SOM. Each node's weight vector is representative of the data points that are mapped to a node. FIG 11 shows two fan diagrams (one for rank and a second one for numerics), which depicts the patterns in the distribution of variables in the data. As mentioned in the previous section, individual fans are representations of the magnitude of each variable in the weight vector. The fan size in each map is based on the values in the *codes* attribute of the SOM object.

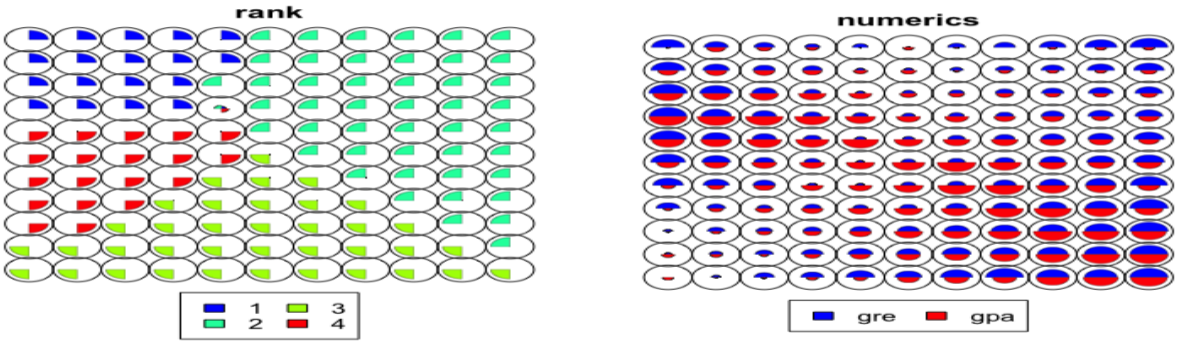

FIG 11: codes map obtained using the new SOM model

In FIG 11, the map on the left shows how the college rank variable, which has four values (1,2,3, and 4) contribute to the codes or the node weights. We can note that the rank variable contributes a maximum of 25% to the node weights. Though the node colors of the left map have shifted compared to the previous SOM's code map, the maximum contribution of the rank variable is still about 25%. Rank value 2 has the most nodes because Rank value 2 is the most occurring value in the data. The map to the right shows the magnitude of the gre and gpa variables to the node weights. We can note that the maximum contribution for each of these variables (gre or gap) to the node weights is 50%, which is same as with the first SOM model. The code map of the numerics has a different node arrangement than the previous SOM model. Like with the previous SOM model, in the map to the right, there are also nodes where numerics have very little contribution to the node weights. This has not changed in the new SOM model.

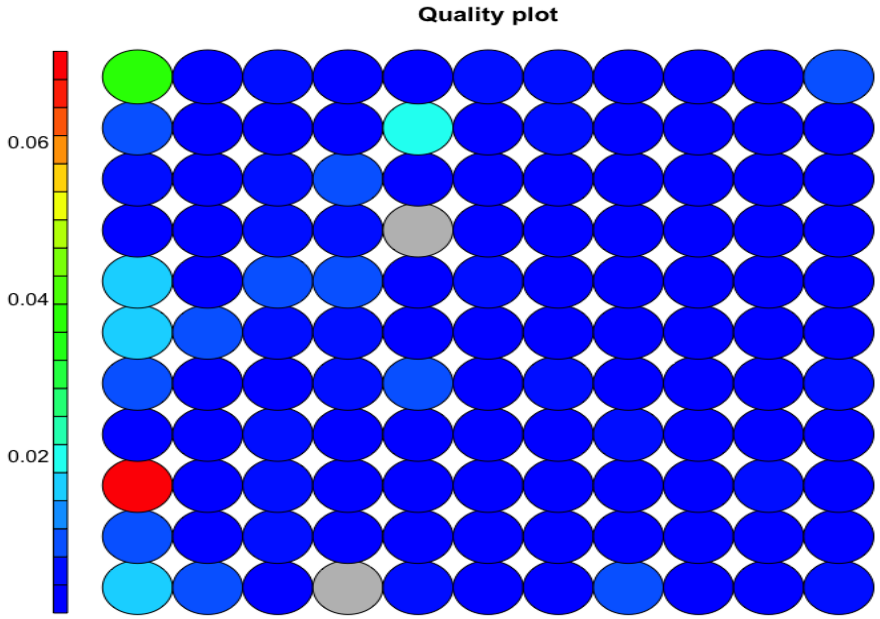

FIG 12: Quality plot for the new SOM model

Lastly, we explored the quality plot (FIG 12) for the new SOM model, which shows the mean distance of objects mapped to a unit to the codebook vector of that unit. The smaller the distances, the better the objects are represented by the codebook vectors. In FIG 12, we see only one red node. So, except for the red node and the empty nodes, most of the code vectors represent the objects well per the quality plot in FIG 12. Compared to the previous quality plot, this quality plot in FIG 12 shows a better quality of the code vectors for the new SOM model.

Section B.5. in the Appendix has the R code for fitting and evaluation of a new SOM model – 11x11 grid size.

8. Reflection of MDS and SOM modeling experiences

For the first part of the assignment to explore the multidimensional scaling, we used the recidivism data set, a random sample of convicts released from prison between July 1, 1977, and June 30, 1978. In addition to the variables about race, alcohol use, drug use, etc., the data set also records duration, which is the time before a convict returned to prison. The aim of using MDS is to uncover any structure or pattern in the proximity matrix. The output of MDS is to create a geometric or spatial configuration in a lower-dimensional space.

For our analysis, we computed the proximity or distance matrix using Euclidean distance inter-point distance measure. To compute this, we used only the continuous or quasi-continuous variables in the data. This is because the metric MDS deals with interval or ratio level data. We used only the variables: number of priors, years of education, number of rule violations in prison, age in months, time served in months, and duration. We removed the nominal variables and ldurat (the log transformation of duration).

We first created a classical multidimensional scaling solution for 2-dimensions using the Euclidean distance dissimilarity matrix. This method is also referred to as principal coordinates analysis due to its similarity to the principal component analysis. The derived coordinate values correspond to the scores of the PCA extracted from the covariance matrix. The goodness-of-fit value is **0.9737015, which shows a good fit**. However, there are negative eigenvalues returned this model. The plot shows two areas of concentration for the points. In terms of interpretation, the dimensions carry a compound meaning involving race, age, type of crime, etc. Next, we used the same dissimilarity matrix to perform non-metric MDS, where the rank order of the proximities is more important than the numerical significance. **For two dimensions, the stress value returned upon convergence is 4.67033. However, the spatial mapping returned by non-metric MDS is exactly the same as the classical MDS.** We next explored the Ramsay MDS solution with the dissimilarity matrix. We selected the monotonic spline transformation. **The stress-1 value returned is 0.3**. Though the solution returned by the Ramsay method has similar spatial structures with two areas of concentrated points, the scale of the dimensions is different. We could not, however, show convergence regions in the mapping.

For the second part of the assignment to explore the Self-Organizing Maps (SOM), we used the college acceptance data set, which is a collection of data of 400 students' acceptance into various college's engineering programs. Our objective of using SOM was to reduce the high-dimensional data to a lower-dimensional nonlinear manifold. SOM maps the projected data to a discrete set of interconnected nodes. Each node represents a grouping or cluster of relatively homogeneous points.

For the analysis, we excluded the response variable (admit) and used only the rank (factor) variable, the gre, and the gpa variables in the data. For the initial round, we select a grid size of 10x10. We selected an Epoch value of 2000. Next, we used the training progress, counts, neighbor distance, codes map, and quality plots to assess the quality of the SOM model. Though the Epoch value is sufficient, we determined a larger grid size may serve better. Based on that, we re-fitted a second SOM model with a grid size of 11x11. We re-assessed the training progress, counts, neighbor distance, codes map, and quality plots of the larger grid size SOM model. **These plots showed that the SOM model with a larger grid size provides better representation for the college acceptance data.**

From this exercise, we learned that MDS's strategy is to keep the most dissimilar data points as far apart as possible. Its focus is to preserve the distances. SOM, on the other hand, keeps the most similar data points together, and the focus is on preserving the neighborhood distances. Overall, between the two methods for dimension-reduction and data visualization techniques, the Self-Organizing Maps is more intuitive and easier to understand than the Multidimensional scaling.

Appendix A – Multidimensional Scaling

A.1. Data Preparation

```
library(readxl)
library(tidyverse)
library(dplyr)

setwd("/Users/harini-mac/Desktop/Northwestern University/MSDS-411/Week5/Assignment03/")
# read the recidivism file
my.data <- read_excel(path="recidivism.xlsx")

# Initial exploration
str(my.data)
head(my.data)
names(my.data)
# check the dimensions of the data
dim(my.data)

#add an ID column to the data
my.data <- tibble::rowid_to_column(my.data, "ID")
# remove any rows with missing values
recidivism.df <- na.omit(my.data)
# check the dimensions again
dim(recidivism.df)
names(recidivism.df)

# Create a label for each row
addLabel <- function(x) {
        label<- x['ID']
        b <- ifelse(x['black'],(label<-paste(label,'bl',sep="_")),'')
        a <- ifelse(x['alcohol'],(label<-paste(label,'al',sep="_")),'')
        d <- ifelse(x['drugs'],(label<-paste(label,'dr',sep="_")),'')
        s <- ifelse(x['super'],(label<-paste(label,'su',sep="_")),'')
        m <- ifelse(x['married'],(label<-paste(label,'ma',sep="_")),'')
        f <- ifelse(x['felon'],(label<-paste(label,'fe',sep="_")),'')
        w <- ifelse(x['workprg'],(label<-paste(label,'wp',sep="_")),'')
        pr <- ifelse(x['property'],(label<-paste(label,'pr',sep="_")),'')
        pe <- ifelse(x['person'],(label<-paste(label,'pe',sep="_")),'')
        c <- ifelse(x['cens'],(label<-paste(label,'cn',sep="_")),'')

        label <- paste(label, toString(x['priors']), 'pri', sep="_")
        label <- paste(label, toString(x['educ']), 'ed', sep="_")
        label <- paste(label, toString(x['age']), 'age',sep="_")
        label <- paste(label, toString(x['tserved']), 'tsrv',sep="_")
        label <- paste(label, toString(x['durat']), 'du',sep="_")

        return(label)
}
lbl <- apply(recidivism.df,1,function(x) addLabel(x))
# Create a label variable for each row in the data set
recidivism.df$label <- lbl
```

A.2. Exploratory Data Analysis (EDA)

```
###############################################################################################
#                          EXPLORATORY DATA ANALYSIS                                          #
###############################################################################################
# obtain counts for the distribution table for each of the nominal variable
cnt_black <- table(recidivism.df$black)
cnt_alcohol <- table(recidivism.df$alcohol)
cnt_drugs <- table(recidivism.df$drugs)
cnt_super <- table(recidivism.df$super)
cnt_married <- table(recidivism.df$married)
cnt_felon <- table(recidivism.df$felon)
cnt_workprg <- table(recidivism.df$workprg)
cnt_property <- table(recidivism.df$property)
cnt_person <- table(recidivism.df$person)
cnt_cens <- table(recidivism.df$cens)
# Create the table with the counts of black, alcohol, drugs, super, married, felon, workprg, property, person, and cens
# varables
tbl <- cbind(cnt_black, cnt_alcohol, cnt_drugs, cnt_super, cnt_married, cnt_felon, cnt_workprg, cnt_property, cnt_person, cnt_cens)
tbl


# Create a histogram of the nominal variables - black, alcohol, drugs, super, married, felon, workprg, property, person, and cens
par(mfrow=c(5,2))
hist(recidivism.df$black,col=c("green","blue"),main="Histogram of the black variable (0=No, 1=Yes)")
hist(recidivism.df$alcohol,col=c("green","blue"),main="Histogram of the alcohol variable (0=No, 1=Yes)")
hist(recidivism.df$drugs,col=c("green","blue"),main="Histogram of the drugs history variable (0=No, 1=Yes)")
hist(recidivism.df$super,col=c("green","blue"),main="Histogram of the supervised variable (0=No, 1=Yes)")
hist(recidivism.df$married,col=c("green","blue"),main="Histogram of the married when incarcerated \n variable (0=No, 1=Yes)")
hist(recidivism.df$felon,col=c("green","blue"),main="Histogram of the felony variable (0=No, 1=Yes)")
hist(recidivism.df$workprg,col=c("green","blue"),main="Histogram of the N.C. prison work program \n variable (0=No, 1=Yes)")
hist(recidivism.df$property,col=c("green","blue"),main="Histogram of the property crime variable (0=No, 1=Yes)")
hist(recidivism.df$person,col=c("green","blue"),main="Histogram of the crime against person \n variable (0=No, 1=Yes)")
hist(recidivism.df$cens,col=c("green","blue"),main="Histogram of the duration right censored \n variable (0=No, 1=Yes)")

# function to obtain the mode of a variable
getmode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}

# EDA for priors variable
par(mfrow=c(4,2))
table(recidivism.df$priors)
tbl2 <- cbind(sum(recidivism.df$priors==0),sum(recidivism.df$priors>0))
colnames(tbl2) <- c("priors=0","priors>0")
tbl2
colors = rainbow(length(unique(recidivism.df$priors)),start=0.1,end=0.5)
hist(recidivism.df$priors,col=colors,main="Histogram of number of prior convictions")
getmode(recidivism.df$priors)
median(recidivism.df$priors)
boxplot(recidivism.df$priors, main = "Box plot of number of prior convictions", xlab = "priors", ylab = "",
      col = "orange", border = "brown", horizontal = TRUE, notch = TRUE)

# EDA for education variable
table(recidivism.df$educ)
colors = rainbow(length(unique(recidivism.df$educ)),start=0.2,end=0.6)
hist(recidivism.df$educ,col=colors,main="Histogram of years of schooling")
getmode(recidivism.df$educ)
median(recidivism.df$educ)
boxplot(recidivism.df$educ,main = "Box plot of education",xlab = "Education",ylab = "",col = "green",
      border = "brown", horizontal = TRUE,notch = TRUE)

# EDA for rules variable
table(recidivism.df$rules)
tbl3 <- cbind(sum(recidivism.df$rules==0),sum(recidivism.df$rules>0))
```

```
colnames(tbl3) <- c("rules=0","rules>0")
tbl3
colors = rainbow(length(unique(recidivism.df$rules)),start=0.4,end=0.9)
hist(recidivism.df$rules,col=colors,main="Histogram of number of rules violations in prison")
getmode(recidivism.df$rules)
median(recidivism.df$rules)
boxplot(recidivism.df$rules, main = "Box plot of number of rules violations in prison",xlab = "number of rules",ylab = "",col = "blue",border
= "brown",
    horizontal = TRUE,notch = TRUE)

# EDA for age variable
table(recidivism.df$age)
hist(recidivism.df$age, main="Histogram for age variable",xlab="age",border="blue", col="lavender", xlim=c(195,940), las=1,breaks=100)
getmode(recidivism.df$age)
summary(recidivism.df$age)
boxplot(recidivism.df$age,main = "Box plot of age",xlab = "age",ylab = "",col = "violet",border = "brown", horizontal = TRUE,notch = TRUE)

par(mfrow=c(4,2))
# EDA for time served variable
table(recidivism.df$tserved)
hist(recidivism.df$tserved,main="Histogram for time served",xlab="tserved", border="blue",col="green",xlim=c(0,220),las=1,breaks=100)
getmode(recidivism.df$tserved)
summary(recidivism.df$tserved)
boxplot(recidivism.df$tserved, main = "Box plot of time served", xlab = "time served", ylab = "",col = "gold",border = "brown",horizontal =
TRUE,notch = TRUE)

# EDA for follow variable
table(recidivism.df$follow)
hist(recidivism.df$follow,main="Histogram for follow up variable",xlab="follow",border="blue",col="green",xlim=c(70,81),las=1,breaks=10)
getmode(recidivism.df$follow)
summary(recidivism.df$follow)
boxplot(recidivism.df$follow,main = "Box plot of follow up", xlab = "time served",ylab = "",col = "darkgreen",border = "brown",horizontal =
TRUE,notch = TRUE)

# EDA for duration variable
table(recidivism.df$durat)
getmode(recidivism.df$durat)
summary(recidivism.df$durat)
hist(recidivism.df$durat, main="Histogram for duration variable",xlab="Duration",border="blue", col="red", xlim=c(1,81),las=1,breaks=80)
boxplot(recidivism.df$durat, main = "Box plot of duration", xlab = "Duration",ylab = "",col = "red",border = "brown",horizontal =
TRUE,notch = TRUE)

# EDA for log duration variable
table(recidivism.df$ldurat)
getmode(recidivism.df$ldurat)
summary(recidivism.df$ldurat)
hist(recidivism.df$ldurat, main="Histogram for log duration variable", xlab="Log duration", border="blue",col="pink",
xlim=c(0,5),las=1,breaks=100)
boxplot(recidivism.df$ldurat, main = "Box plot of log duration",xlab = "log duration",ylab = "",col = "purple",border = "brown",horizontal =
TRUE,notch = TRUE)
```

A.3. Dissimilarity Matrix using Euclidean distances

```
################################################################################################
#                          DISSIMILARITY MATRIX USING EUCLIDEAN DISTANCES                      #
################################################################################################

# Create a reduced data set with the variables - priors, educ, rules, age, tserved, durat
# Use label variable as the row names
reduced.set<- recidivism.df %>% select(-ID, -black, -alcohol, -drugs, -super, -married, -felon, -workprg, -property, -person, -cens, -follow, -
ldurat, -label)
.rowNamesDF(reduced.set,make.names=TRUE) <-recidivism.df$label
row.names(reduced.set)
d1 <- dist(reduced.set) # euclidean distances between the rows
d1 <- data.matrix(d1)
#heatmap with no row or column reordering
heatmap(d1, Colv = NA, Rowv = NA, scale="none")
#heatmap with reordering and hierarchical clustering
heatmap(d1, scale="none")
```

A.4. Classical Multidimensional Scaling

```
#############################################################################################
#                          CLASSICAL MDS                                                    #
#############################################################################################
fit <- cmdscale(d1, eig=TRUE, k=2)
fit
par(mfrow=c(1,1))
plot(fit$eig,main="Eigen value plot")
x <- fit$points[,1]
y <- fit$points[,2]
plot(x, y, xlab="Dimension 1", ylab="Dimension 2",main="Metric MDS",type="n")
text(x, y, labels = row.names(reduced.set), cex=.7,col='blue')
```

A.5. Non-metric Multidimensional Scaling

```
#############################################################################################
#                          Non-metric MDS                                                   #
#############################################################################################
# Nonmetric MDS
# N rows (objects) x p columns (variables)
# each row identified by a unique row name

library(MASS)
plot(rank(as.matrix(d1)), as.matrix(d1))
fit <- isoMDS(d2, k=2) # k is the number of dim
fit # view results
# plot solution
x <- fit$points[,1]
y <- fit$points[,2]
plot(x, y, xlab="Dimension 1", ylab="Dimension 2",
   main="Nonmetric MDS", type="n")
text(x, y, labels = row.names(reduced.set), cex=.7,col='green')
```

A.6. Ramsay's method for Multidimensional Scaling

```
##############################################################################################
#                                          Ramsay MDS                                        #
##############################################################################################
# Need smacof package
library(smacof)
# use mds function to create model using mspline transformation
fit.ramsay <- mds(d1,type="mspline",ndim=2)
fit.ramsay
# plot solution
plot(fit.ramsay, xlab="Dimension 1", ylab="Dimension 2",
   main="Ramsay MDS plot",col="red",cex=0.9)

# Did not work - confEllipse never completes
#conf.ramsay <- confEllipse(fit.ramsay)
#plot(conf.ramsay, eps = 0.01, ylim = c(-100, 100), ell = list(lty = 1, col = "blue"))
```

Appendix B – Self-Organizing Maps

B.1. Data preparation and helper functions

```
require(tidyverse)
require(kohonen)
require(ggplot2)
require(ggridges)


############################
# Resources
############################
# https://rpubs.com/erblast/SOM

############################
# Helper functions
############################

# find the graph node number by the coordinates
find_node_by_coordinates <- function(x, y, grid_width) {
  return(((y * grid_width) + x) - grid_width)
}

# return the number of observations represented by each node
get_node_counts <- function(x) {
  df <- data.frame(node = x)
  counts <- df %>%
    group_by(node) %>%
    summarize(observations = n())
}

# guideline for grid size = 5 * sqrt(N)
# where N is the number of observations in the data set
find_grid_size <- function(N) {
  return(floor(sqrt(sqrt(N) * 5)))
}

# Shane Lynn 14-01-2014 used to define the palette
coolBlueHotRed <- function(n, alpha = 1) {
  rainbow(n, end=4/6, alpha=alpha)[n:1]
}

############################
# Data load and prep
############################

# load data - select the college_acceptance.csv
myFile <- file.choose()
raw_data <- read.csv(myFile,header=TRUE)
names(raw_data)
str(raw_data)

# check for missing observations
# remove any rows with missing values
raw_data <- na.omit(raw_data)
dim(raw_data)
```

B.2. Exploratory Data Analysis

```
###############################################################################################
#                          EXPLORATORY DATA ANALYSIS                                          #
###############################################################################################

# EDA for the admit variable
table(raw_data$admit)

par(mfrow=c(3,2))
# EDA for the gre variable
colors = rainbow(length(unique(raw_data$gre)),start=0.1,end=0.5)
hist(raw_data$gre,col=colors,main="Histogram of gre")
summary(raw_data$gre)
boxplot(raw_data$gre, main = "Box plot of number of gre", xlab = "gre", ylab = "",
      col = "blue", border = "brown", horizontal = TRUE, notch = TRUE)

# EDA for the gpa variable
colors = rainbow(length(unique(raw_data$gpa)),start=0.1,end=0.5)
hist(raw_data$gpa,col=colors,main="Histogram of gpa")
summary(raw_data$gpa)
boxplot(raw_data$gpa, main = "Box plot of number of gpa", xlab = "gpa", ylab = "",
      col = "magenta", border = "brown", horizontal = TRUE, notch = TRUE)

# EDA for the rank variable
colors = rainbow(length(unique(raw_data$rank)),start=0.1,end=0.5)
barplot(table(raw_data$rank),col=colors,main="Barplot of rank")
summary(raw_data$rank)
table(raw_data$rank)
boxplot(raw_data$rank, main = "Box plot of number of rank", xlab = "rank", ylab = "",
      col = "purple", border = "brown", horizontal = TRUE, notch = FALSE)

raw_data$admit <- as.factor(raw_data$admit)
levels(raw_data$admit) <- c('Not Admitted', 'Admitted')

# review the distributions of not admitted and admitted students
raw_data %>%
 gather(variable, value, -admit) %>%
 ggplot(aes(y = as.factor(variable),
         fill = admit,
         x = percent_rank(value))) +
 geom_density_ridges() +
 ggtitle('Admit Distributions') +
 theme(plot.title = element_text(hjust = 0.5)) +
 xlab('Value') +
 ylab('Variable')

# remove admit and rank
reduced_set <- raw_data[, 2:3]
# center and scale the data
data <- as.matrix(scale(reduced_set, center=TRUE, scale=TRUE))
data <- as.data.frame(data)

# Add the rank variable as a factor variable
data <- cbind(data, raw_data[,4])
colnames(data) <- c("gre","gpa","rank")
data$rank <- as.factor(data$rank)
```

B.3. Fitting a SOM model – 10x10 grid size

```
################################################################################
#                          FITTING A SOM MODEL                                 #
################################################################################

# find all columns having numeric data
numerics <- data %>%
 select_if(is.numeric) %>%
 names

# find all columns having factors
factors <- data %>%
 select_if(is.factor) %>%
 names

# initialize the data_list and distances vector
data_list = list()
distances = vector()

# create a layer for each factor
for (fac in factors){
 data_list[[fac]] = kohonen::classvec2classmat( data[[fac]] )
 distances = c(distances, 'tanimoto')
}
# create a layer for numerics
data_list[['numerics']] = scale(data[,numerics])
distances = c(distances, 'sumofsquares')

# review data_list
str(data_list)
names(data_list)

# review distance measures
distances

# call the find_grid_size function to calculate the grid size
map_dimension = find_grid_size(dim(raw_data)[1])

# set the number of times the model will cycle through the observations
epochs = 2000
set.seed(123)

# create a grid onto which the som will be mapped
som_grid = somgrid(xdim = map_dimension
                   ,ydim = map_dimension
                   ,topo = "rectangular")

# train the SOM
cc_som = supersom(data_list
                  ,grid = som_grid
                  ,rlen = epochs
                  ,alpha = c(0.1, 0.01)
                  ,whatmap = c(factors, 'numerics')
                  ,dist.fcts = distances
                  ,keep.data = TRUE
)
```

B.4. Evaluation of SOM model

```
##############################################################################################
#                          EVALUATION OF THE SOM MODEL                                       #
##############################################################################################

par(mfrow=c(1,1))
# Traning Progress plot
plot(cc_som, type = 'changes')

# Counts plot
plot(cc_som, type = "counts", palette.name = coolBlueHotRed)

# Observations by node
cc_som$unit.classif
observations_by_node <- get_node_counts(cc_som$unit.classif)

# Neighbor distance plot
plot(cc_som, type = "dist.neighbours", palette.name = coolBlueHotRed)

# Evaluate the anomalies
anomalies <- find_node_by_coordinates(1,3,10)
rows <- which(cc_som$unit.classif == anomalies)
raw_data[rows, ]

anomalies <- find_node_by_coordinates(7,8,10)
rows <- which(cc_som$unit.classif == anomalies)
raw_data[rows, ]

anomalies <- find_node_by_coordinates(8,7,10)
rows <- which(cc_som$unit.classif == anomalies)
raw_data[rows, ]

# Codes map plot
par(mfrow=c(1,2))
plot(cc_som, type = "codes", palette.name = coolBlueHotRed)

#obtain the codes for the rank variable
cc_som$codes[["rank"]]


#plot(cc_som, type = "property", property = getCodes(cc_som)[["rank"]],
#main=colnames(getCodes(cc_som))[["rank"]], palette.name=coolBlueHotRed)
# quality plot
par(mfrow=c(1,1))
plot(cc_som, type = "quality", palette.name = coolBlueHotRed)
```

B.5. New SOM model – 11x11 grid size

```
#############################################################################################
#                New SOM Model - grid size 11x11                                            #
#############################################################################################

# use a grid size of 11x11
map_dimension = 11

# set the number of times the model will cycle through the observations
epochs = 2000
set.seed(123)

# create a grid onto which the som will be mapped
som_grid = somgrid(xdim = map_dimension
                   ,ydim = map_dimension
                   ,topo = "rectangular")

# train the SOM
cc_som = supersom(data_list
                  ,grid = som_grid
                  ,rlen = epochs
                  ,alpha = c(0.1, 0.01)
                  ,whatmap = c(factors, 'numerics')
                  ,dist.fcts = distances
                  ,keep.data = TRUE
)

par(mfrow=c(1,1))
# Training progress plot
plot(cc_som, type = 'changes')
# counts plot
plot(cc_som, type = "counts", palette.name = coolBlueHotRed)

# number of observations by node
cc_som$unit.classif
observations_by_node <- get_node_counts(cc_som$unit.classif)

# neighbor distance plot
plot(cc_som, type = "dist.neighbours", palette.name = coolBlueHotRed)

# obtain the anomalies
anomalies <- find_node_by_coordinates(5,11,11)
rows <- which(cc_som$unit.classif == anomalies)
raw_data[rows, ]

anomalies <- find_node_by_coordinates(5,9,11)
rows <- which(cc_som$unit.classif == anomalies)
raw_data[rows, ]

anomalies <- find_node_by_coordinates(1,3,11)
rows <- which(cc_som$unit.classif == anomalies)
raw_data[rows, ]

par(mfrow=c(1,2))
# obtain the codes plot
plot(cc_som, type = "codes", palette.name = coolBlueHotRed)

#view the codes for the rank variable
cc_som$codes[["rank"]]

#plot(cc_som, type = "property", property = getCodes(cc_som)[["rank"]],
#main=colnames(getCodes(cc_som))[["rank"]], palette.name=coolBlueHotRed)
par(mfrow=c(1,1))
#quality plot
plot(cc_som, type = "quality", palette.name = coolBlueHotRed)
```