Unsupervised Learning: Project 4

Florence Hugh, August Jonasson, Sofia Näslund

October 2024

Introduction

The aim of this project is to reduce the dimensionality and, if possible, cluster data related to phishing. Phishing is a type of cyberattack where attackers trick individuals into providing sensitive information, such as passwords, credit card numbers, or personal data, by posing as a trustworthy entity, often through deceptive emails, websites, or messages. The dataset we have chosen specifically concerns URLs and was selected due to our personal interest in the subject and its increasing relevance in daily life.

Description of the data set

The dataset, obtained from Kaggle.com, contains 18 features for 2.5 million URL observations across various websites. We decided to remove several of these features for different reasons. "URL," "source," and "who_is_data" were removed: "URL" because it serves as the identifier for each observation, and the other two because they were likely used to generate other features. Additionally, the "domain_age_days" feature was excluded due to a large number of missing values. This leaves us with a total of 14 remaining features - of which 5 were categorical (binary) and 9 quantitative (integers or reals).

From these 14 features we sampled only 1500 observations (URLs), as this gave our scripts reasonable running times. Table 1 provides a summary of the remaining features.

Binary Features	**Explanation**
`"starts_with_ip"`	Indicates if the URL starts with an IP address
`"has_punycode"`	Indicates if the URL contains punycode
`"domain_has_digits"`	Indicates if the domain contains digits
`"has_internal_links"`	Indicates if the URL contains internal links
`"label"`	Indicates whether the link is legitimate or phishing
Integer Features	**Explanation**
`"url_length"`	Number of characters in the URL
`"dot_count"`	Number of dots ('.') in the URL
`"at_count"`	Number of at:s ('@') in the URL
`"dash_count"`	Number of dashes ('-') in the URL
`"tld_count"`	Number of top-level domains in the URL
`"subdomain_count"`	Number of subdomains in the URL
Continuous Features	**Explanation**
`"url_entropy"`	Randomness of the URL characters
`"nan_char_entropy"`	Randomness of non-alphanumeric characters in the URL
`"digitletter_ratio"`	Ratio of digits to letters in the URL

Table 1: Features in the dataset

Without going into detail in this report - as this project mainly should be about dimensionality reduction and clustering - we mention here that all of our non-categorical features have been centered and normalized. We did this as this was a requirement for the dissimilarity measure of our choice.

Dissimilarity measure

Since the data we are using is of the mixed feature type - categorical and numerical - we are really only left with one choice: the Gower's distance; which defines the dissimilarity between object i and object j as

$$d_{ij} = \frac{\sum_k^D w_{ijk} d_{ijk}}{\sum_k^D w_{ijk}},$$

where D is the number of features, w_{ijk} are the weights (all set to 1 in our case) and d_{ijk} is the specific feature dissimilarity between objects i and j (Gower 1971). For our binary features, $d_{ijk} = 0$ when the k:th feature of object i equals the k:th feature of object j and $d_{ijk} = 1$ otherwise. For the numerical features , the dissimilarity is defined as

$$d_{ijk} = \frac{|y_{ik} - y_{jk}|}{R_k},$$

where y_{ik} is the k:th feature of the i:th data point and R_k is the range of the k:th feature, thus ensuring that $0 \leq d_{ijk} \leq 1$.

As such, we can interpret the Gower's distance between y_i and y_j as a sort of mean feature dissimilarity between the objects.

Exploratory Analysis

Having chosen our dissimilarity measure, we can start exploring the data from the perspective of the unsupervised learning toolbox.

The first step to our exploration will be hierarchical clustering. With this we hope to intuit early on if there might be any natural clusters in our data. We tried all three inter-cluster dissimilarity measures but as the single-linkage result lead to heavy chaining and as the group-average was hard to interpret we only show the complete-linkage result here (Hastie et al. 2009, p. 523).

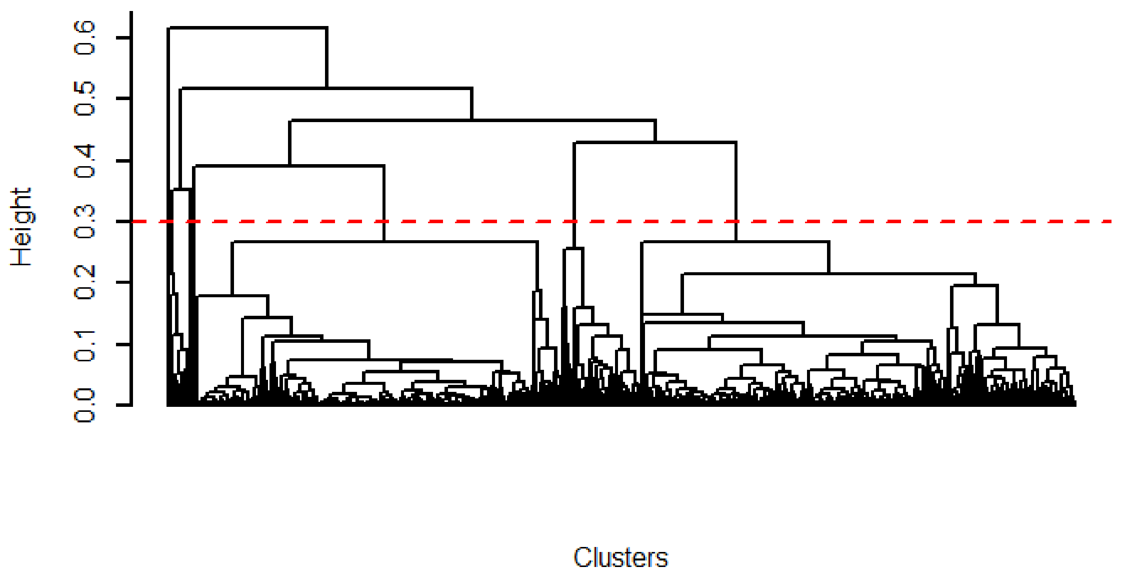

Figure 1: Hierarchical clustering with complete linkage

From Figure 1 we see the resulting dendrogram from using the `hclust` function in R (Doe 2024). The dashed red line represents what we believe to be a decent cut-off threshold - as we are sufficiently high up in the dendrogram such that a substantial amount of points are being merged, and simultaneously cutting through some of the tallest merges, indicating separation of the data. From this we intuit that there seem to be some natural cluster structure to our data and that the amount of clusters might be between three and seven (based on the number of merges that are being cut by the red line). As high as seven seems unlikely though, as some of the merges to the far left in the plot seem to involve only a single or maybe a few data points.

Another tool we can use in order to better grasp the structure of our data is a self-organizing map. This can be computed using the `kohonen` package (Wehrens and Kruisselbrink 2023).

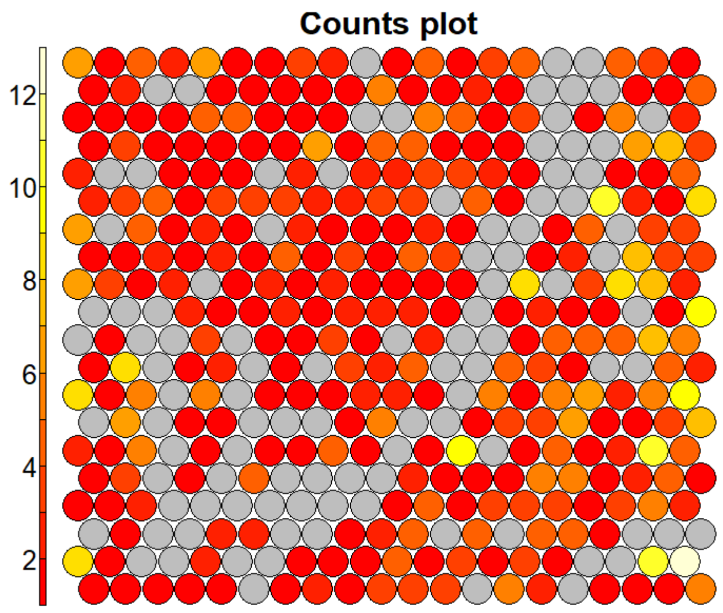

Figure 2: Self Organizing Map

The Self Organizing Map in Figure 2 helps with visualisation by organizing the data set onto a 2D space while preserving the topological relationships, to easier observe relationships within the data. The algorithm works by moving prototypes in a rectangular grid closer to the data points, while preserving the spatial relationships between them. (Hastie et al. 2009, p. 528-529). In this case we use the hexagonal grids for better visual observation with 400 hexagonal nodes, performed using the default settings for α and r. Each hexagonal shows the number of data points mapped to each grid, where empty units are gray (Wehrens and Kruisselbrink 2023, p.14, 20).

The map shows that there are four separated clusters, where the gray-colored cells delimits them. There are two broader clusters on top of the map, where the left one have less mapped objects mapped to the unit (lower density) and the right one have partially higher density in some areas. In the bottom corners there are two clusters with higher density but less data points.

As with the hierarchical clustering, we can see here too that there seem to be some natural separation to our data.

Dimensionality reduction

When choosing a method for dimensionality reduction there are broadly speaking two main aspects to consider: the preservation of distances and the preservation of topology. Since the original space of our data is mixed between categorical and numerical, focusing too much on the preservation of distances might not make a lot of sense. For example, consider two cars: one red and one blue; the first one travelling at 70km/h and the second one at 100km/h. It is easy to quantify the difference in speed between the two cars, but simultaneously quantifying the difference in color doesn't make a lot of sense. This, however, is metaphorically speaking precisely what we have done, using the Gower's distance. As we believe this dissimilarity doesn't make a lot of sense as a distance (in its usual meaning) we want to relax the distance-preservation if possible, i.e. make as few assumptions as possible on the original space.

Now, as our original space is definitely not Euclidean, and neither is our dissimilarity measure, we are really only left with two options: metric multidimensional scaling (mMDS) and non-metric multidimensional scaling (nmMDS). The one of the two that assumes the least is the latter, and as such we choose nmMDS for our attempt at reducing the dimensionality.

Non-metric MDS still preserves distances but through the ranking of the distances rather than through the distances directly - somewhat relaxing the assumptions on the original space (Lee, Verleysen, et al. 2007, p.81), while still preserving topology. To perform the non-metric MDS the `isoMDS` function (Ripley 2024, et al. p. 70) was used.

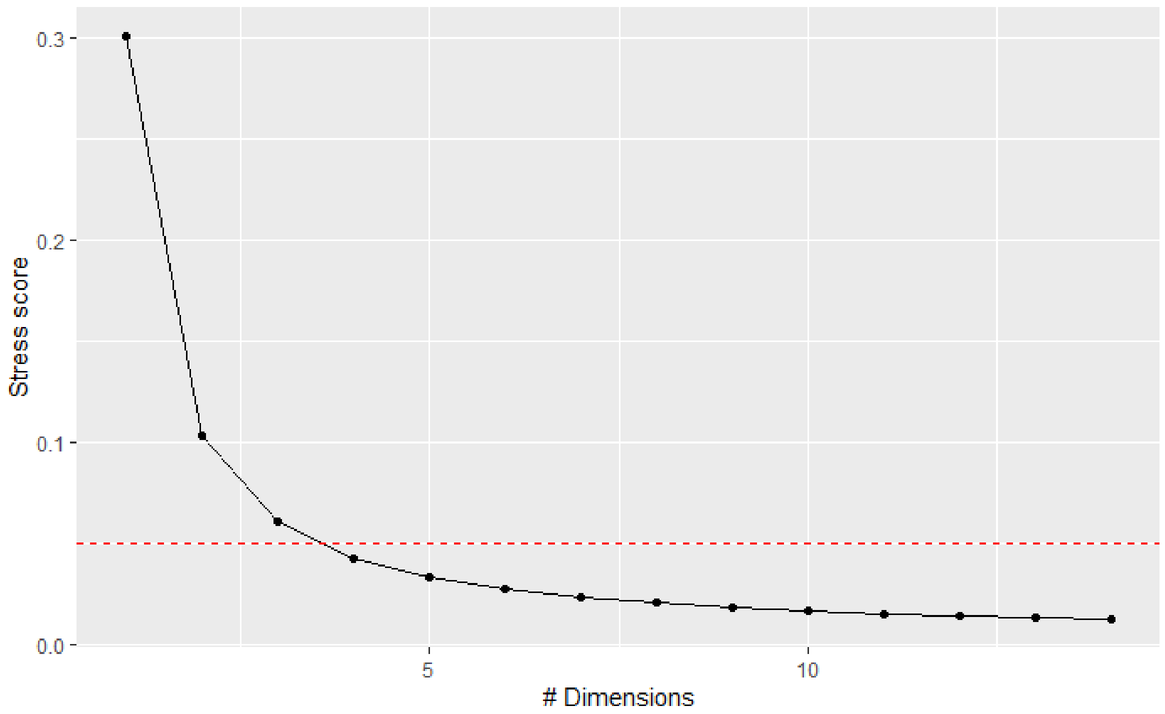

Figure 3: Stress scores of non-metric MDS dimensionality reduction

When performing non-metric MDS, we measure the goodness of fit of the reduction with the help of a stress score plot, which can be observed in Figure 3. A threshold of 0.05 was set, as stress values below this are considered excellent (Watson 2014). Here we can observe that the four-dimensional optimization falls below the threshold, and therefore is a good fit.

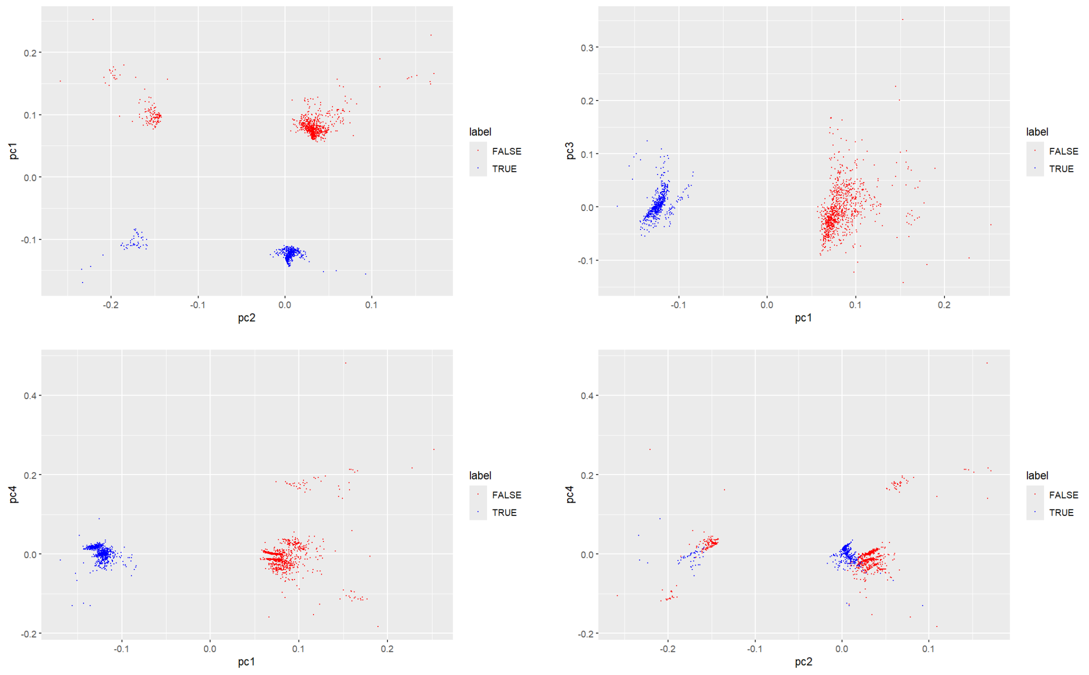

Figure 4: Non-metric MDS pairs of principal coordinates

In Figure 4, a select amount of pairwise principal coordinates (PCs) from the optimization of the nm-MDS stress function (#dim = 4) are shown. The phishing URLs are the red points and the legitimate are blue. Observation of the first two PCs (top left) shows four clearly separated clusters. Separation between legitimate URLs and phishing URLs is also evident from this plot. When plotting the first eigenvector against the third and fourth there seems to be only two clusters, which are separated for the true and false websites. When plotting the second and fourth vectors against each other it seems as the four clusters seen with the first two vectors now have collapsed into two clusters, where the true and false data points are connected. This observation further strengthens our assumption of four clusters within the dataset.

Dimensionality reduction validation

In order to assess the quality of the dimensionality reduction, we have to validate our results. As we have already considered the stress score of the non-metric MDS when choosing the dimensionality of our projected space, what is left to do is to study how well the topology is preserved through the projection. As such we turn to the co-ranking matrix, created by using the `coRanking` package (Kraemer 2024).

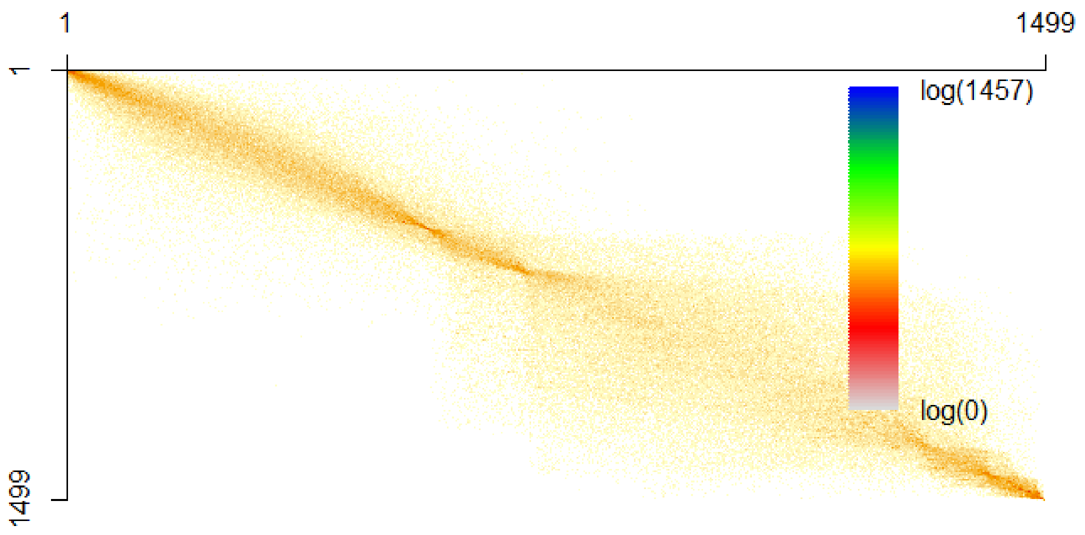

Figure 5: Co-ranking matrix illustrating intrusions/extrusions of the non-metric MDS dim. reduction

From Figure 5 we see the imaged result of the co-ranking matrix. First, we note that the matrix is sparse. This means that the overall topology preservation of the dimensionality reduction at least isn't terrible. For the low ranks (the nearest neighbors) of the data points, intrusions and extrusions are very mild (high concentration around main diagonal) which tells us that local distance ranks are well preserved. This in turn already tells us that the topology is rather well preserved since local preservation matters more than global (think of the swiss roll unfolding).

We do see some less mild intrusions and extrusions for the mid to mid-to-high ranks of the neighbors which somewhat messes up the structure of our data. However, since our goal is clustering, where not messing up the separation of data is essential, the fact that the far-away neighbors are still far away (mild intrusion/extrusion) tells us that we have not messed up the separation of the data. We thus believe that our dimensionality reduction has worked out quite well. For references on how to interpret the co-ranking matrix, look to Li 2024 lecture slides on validation.

Clustering

After having reduced the number of dimensions in our data, our next step is to choose a suitable clustering method. In order to do this we have to consider aspects such as cluster shapes, cluster densities, how well separated the clusters are, and how we want to deal with outliers and noise. Starting with the noise aspect, we have decided not to remove any. The motivation for this is that the data we are analysing only represents a very small fraction of the original data (0.006%), and what seems to be noise may in fact come from some very real, but underrepresented, cluster structures (more on this in the discussion section of the report). At the same time, we want outliers to have as little effect as possible on the actual clustering. We want to keep the noise while at the same time minimizing its effect on the clustering method.

Now considering shapes, densities and the separation of clusters we can look to Figure 4 (top left) to build some intuition. The clusters seem to be well-separated, have somewhat varying densities, and their shapes seem a bit non-linear. Cluster centers are far apart and cluster centers have high density. Considering we also want to keep the noise, we land on using the method described in Rodriguez and Laio 2014 (what we in class have referred to as "density peak clustering" (DPC)). In R, we use the package `densityClust` (Pedersen, Hughes, and Qiu 2024) which builds directly on the content of this article. We will now present the results of this method.

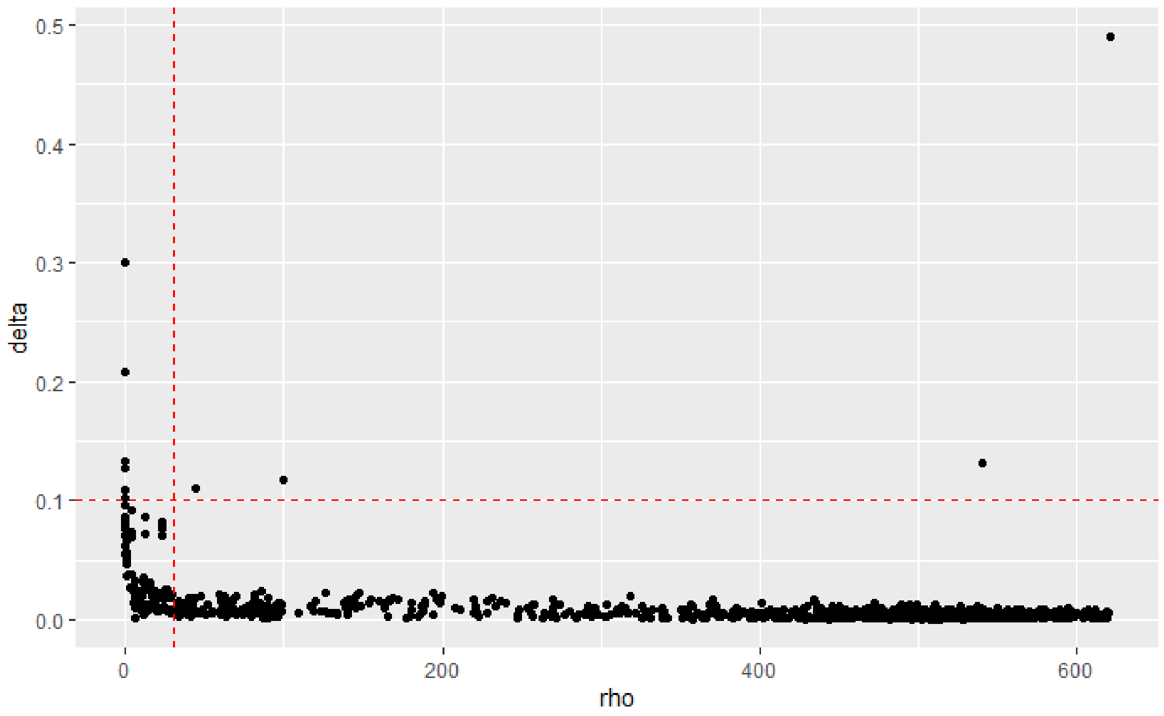

Figure 6: DPC decision graph with dashed lines for $\rho = 30$ and $\delta = 0.1$

A decision graph is constructed by measuring the local density (ρ) of each data point, and its minimum distance to any point with higher density (δ) (Rodriguez and Laio 2014). To set the local density one must decide a value for the cutoff distance (d_c). We tried to follow the recommendation of Rodriguez and Laio 2014 to select d_c such that the average number of neighbors are around 1-2 % of the total data points. In Figure 6 we can observe the decision graph with boundaries $\rho = 30$ and $\delta = 0.1$, resulting in what visually seems to be an appropriate amount of clusters. Letting the dashed lines vary over different values on ρ and δ we can try out all distinct clusterings that result from the DPC method.

After finding the cluster centers, the clusters are formed by assigning the remaining points to its nearest neighbor of higher density (Rodriguez and Laio 2014).

Clustering validation

In order to decide on the appropriate amount of clusters in a valid way we turn to the silhouette coefficients that are used in clustering validation. A high coefficient value means a good fit. Plotting the mean of the silhouette values of all data points against different number of clusters, and then picking out the maximum, is an internal measure of choosing an appropriate amount of clusters (Li 2024, lecture on validation).

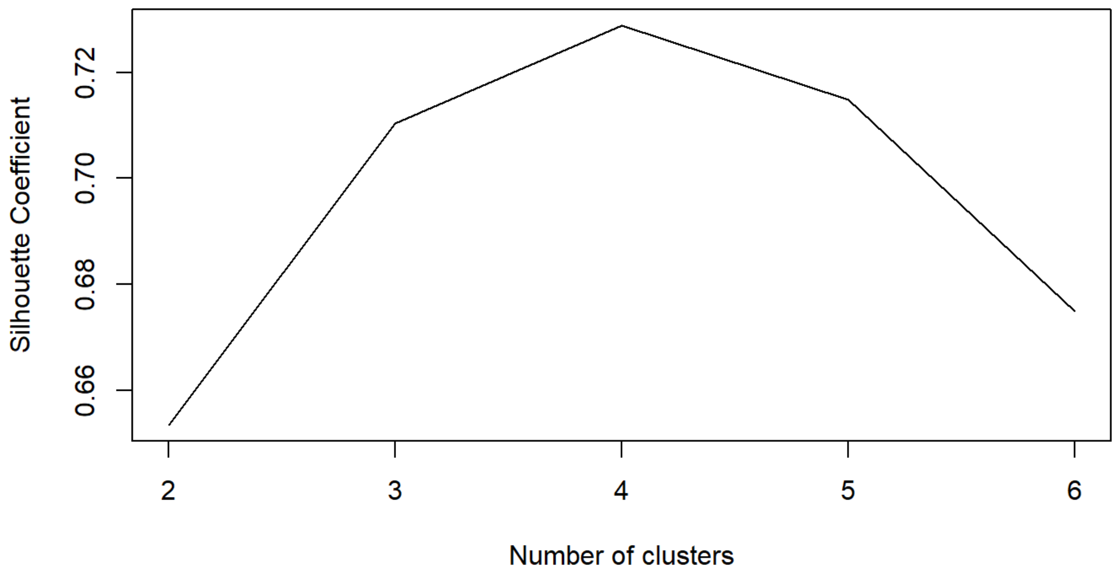

Figure 7: Validation of DPC by the number of clusters and their silhouette score

,

This plot is illustrated in Figure 7 and the elbow confirms that four clusters is appropriate. Hence, we fix $\rho = 30$ and $\delta = 0.1$ as in Figure 6 and cluster our data accordingly. Figure 8 is the exact same plot as Figure 4, now with four-cluster DPC applied, with satisfying result.

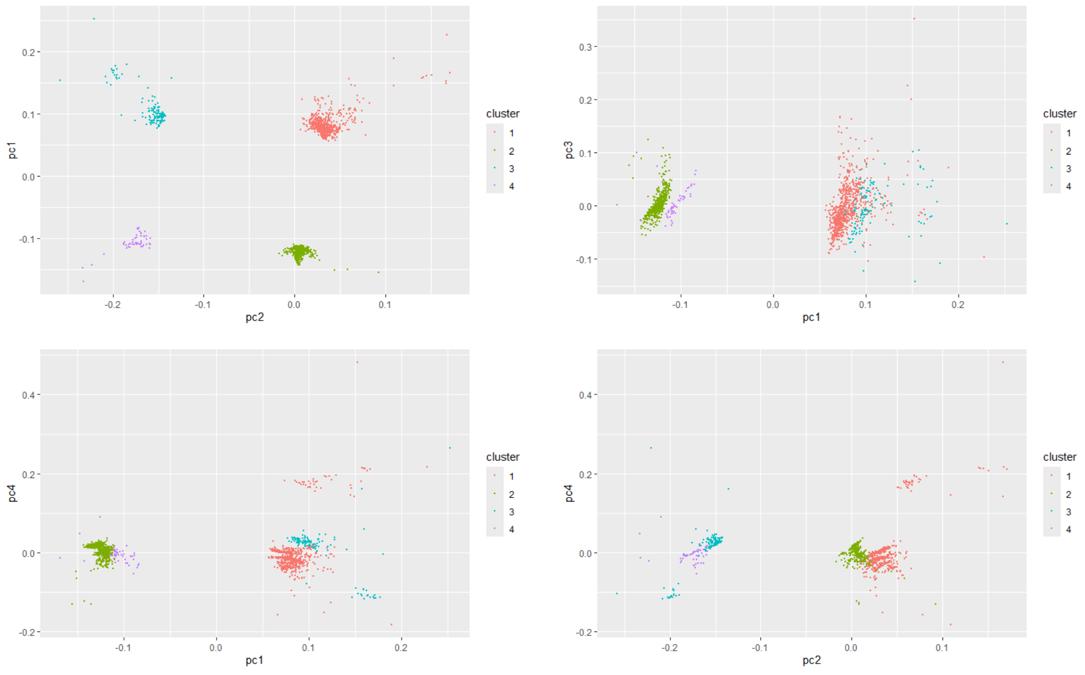

Figure 8: Clustering results

Further validation of the case with four clusters can be observed in Figure 9. The silhouette values of the individual clusters are ranging between $0.63 - 0.81$ with a minimum of above 0.2 for point-wise values. It is further observed that there are no negative silhouette values in neither of the clusters, which means that there are no overlapping of the clusters. In this case where we have high silhouette values both point wise, cluster wise and globally, it is therefore established that the clustering performed on this sample is valid.

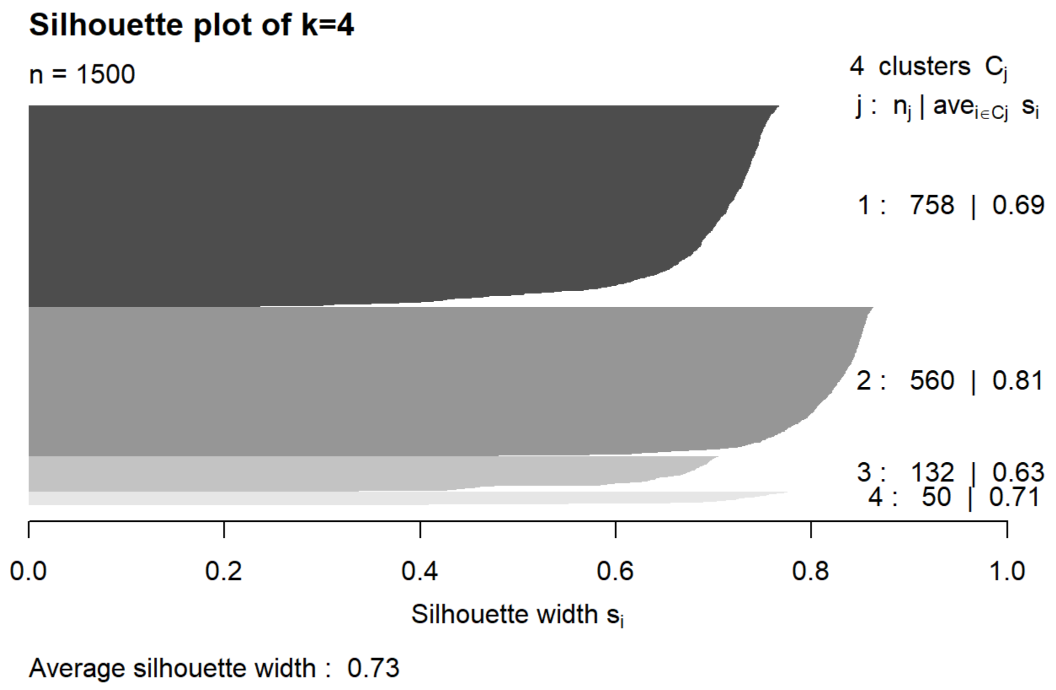

Figure 9: Silhouette plot with four clusters

Discussion

The final result showed that our data could be separated into four clusters, but we need to be reminded that we only considered a small sample of the whole data. This means that the clustering might have looked different if we increased the sample size. For example, in Figure 10 where we took a larger sample of the original data, one can see that the regions from Figure 4 with points looking like outliers could actually be their own clusters or merged into existing ones. Hence why we decided to not remove any “outliers”.

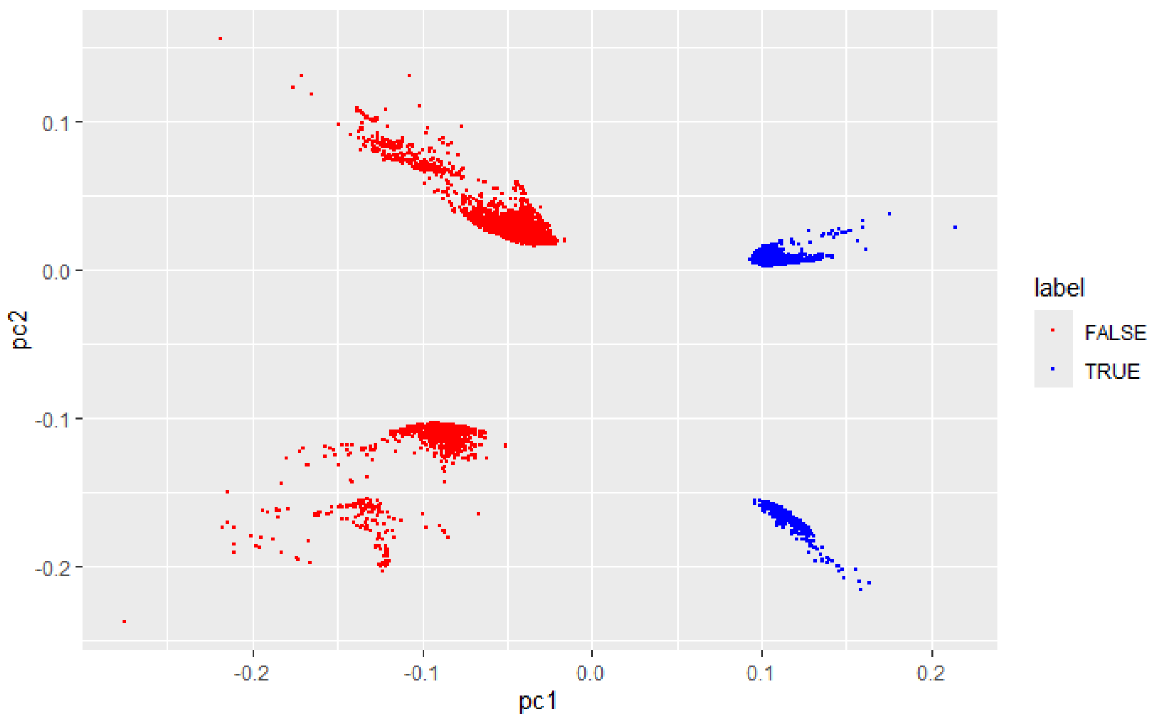

Figure 10: Non-metric MDS on a sample ten times larger than the one used in the analysis

This was the reason why we decided to use DPC rather than DBSCAN, even if it might have worked for our original sample. DBSCAN would have assigned more points as noise whereas DPC considers all points to be included in some cluster.

CTD embedding/Graph PCA

We also tried the CTD embedding (which can be found in the code appendix) but this resulted in clusters consisting only of a single point, i.e. the noise in our data were given too much space. As we wanted to avoid dealing with noise, we didn't use this method.

Conclusion

In conclusion, our sample of the data could successfully be reduced from 14 to 4 dimensions, and then clustered into four clusters. The four clusters are well separated with two groups of phishing data, and two groups with non-phishing data.

For the whole dataset there might however be more clusters, as the noise in the sample could potentially be other clusters that are not visible with a small sample. To observe this however, a stronger computer is needed in order to execute the calculations.

R Code

~~~
```{r, message=FALSE}
library(MASS)
library(StatMatch)
library(ggforce)
library(coRanking)
library(cccd)
library(igraph)
library(kohonen)
library(tidyverse)
library(magrittr)
library(densityClust)
library(cluster)
library(stats)
~~~

```{r}
data_raw <- read_csv("out.csv")
```

```{r}
# this code chunk is for removing unnecessary data and standardizing
set.seed(1995)
larger_sample <- data_raw[sample(nrow(data_raw), 15000), ]

larger_sample <- larger_sample %>%
  select(!c(url, source, whois_data, domain_age_days)) %>%
  mutate(label = ifelse(label == "legitimate", TRUE, FALSE))

larger_sample_num <- larger_sample %>%
  select(!c(label, starts_with_ip, has_punycode,
           domain_has_digits, has_internal_links)) %>%
  scale(scale = TRUE)

larger_sample_cat <- larger_sample %>%
  select(c(label, starts_with_ip, has_punycode,
           domain_has_digits, has_internal_links))

larger_sample <- cbind(larger_sample_num, larger_sample_cat) %>%
  distinct()
```

```{r}
# calculating distances
larger_sample_dis <- gower.dist(data.x = larger_sample)
```

```{r}
# first and second PC based on non-metric
nmMDS_larger_sample <- isoMDS(as.dist(larger_sample_dis), k = 2)
```

```{r}
# first and second pc plot absed on larger sample
nmMDS_larger_sample[[1]] %>%
  data.frame() %>%
  cbind(larger_sample$label) %>%
```

```
  set_colnames(c("pc1", "pc2", "label")) %>%
  ggplot(aes(x = pc1, y = pc2, color = label)) +
    scale_color_manual(values = c("red", "blue")) +
    geom_point(size = 0.1)
```

```{r}
# the dissimilarity matrix based on Gower distance
dis_matrix <- gower.dist(data.x = df_standard)
```

```{r}
# some data that we already computed and reused in order to keep consitency
load("sample_data.Rda")
load("df_standard.Rda")
load("dis_matrix.Rda")
load("all_mds_and_stress.Rda")
```

```{r}
# complete linkage hierarchical clustering
hier_clust <- hclust(as.dist(dis_matrix))
hier_clust_av <- hclust(as.dist(dis_matrix), method = "average")

hier_clust_sing <- hclust(as.dist(dis_matrix), method = "single")

plot(hier_clust, labels = FALSE, hang = -1, main = "", xlab = "Clusters", ylab = "
    Height", sub = "", col = "black",       # Change the color of the dendrogram lines
     lwd = 2)
abline(h=0.3, col = "red", lty= 2, lwd= 2)
```

```{r}
# non-metric MDS (stress score and projection) for all possible #dimensions
mds <- map(seq(1,14), ~isoMDS(as.dist(dis_matrix), k=.x))
```

```{r}
# compiling the stress scores from the non-metric MDS
stress <- map_dbl(seq(1,14), ~ mds[[.x]][[2]])
stress_vs_dims <- data.frame("dimensions" = seq(1,14),
                             "stress_score" = stress/100)

# non-metric MDS stress against dimensions plot
stress_vs_dims %>%
  ggplot(aes(x = dimensions, y = stress_score)) +
  geom_line() +
  geom_point() +
  xlab("# Dimensions") +
  ylab("Stress score") +
  geom_hline(yintercept = 0.05, linetype = "dashed", color = "red")
```

```{r}
# creating the elbow plot for the dimensionality reduction (nmMDS)
elbow_mds <- mds[[4]][[1]] # 4-dim non-metric mds result
```

```
mds_dis <- gower.dist(data.x = elbow_mds)
```

```{r}
# corank matrix for validating the dim reduction
corank_matrix <- coranking(dis_matrix, mds_dis,
                           input_Xi = "dist", input_X = "dist")
```

```{r}
# plotting the corank matrix
imageplot(corank_matrix, main = "")
```

```{r}
# self-organizing map

#Creates a grid
data_grid <- somgrid(xdim = 20, ydim = 20, topo = "hexagonal")

# the SOM funktion
SOM_model <- som(X = dis_matrix, grid = data_grid, keep.data = TRUE)

#Plot of the SOM counts
plot(SOM_model, type = "counts")
```

```{r}
# density peak clustering (DPC)
clustering <- densityClust(as.dist(mds_dis), dc=0.045, gaussian = FALSE)
plot(clustering)

clustering6 <- findClusters(clustering, rho=20, delta=0.078) # 6cluster
clustering5 <- findClusters(clustering, rho=20, delta=0.08) # 5cluster
clustering4 <- findClusters(clustering, rho=20, delta=0.1) # 4cluster
clustering3 <- findClusters(clustering, rho=50, delta=0.1) # 3cluster
clustering2 <- findClusters(clustering, rho=500, delta=0.1) # 2cluster
plot(clustering4)
```
```{r}
df_clust <- data.frame(rho = clustering$rho,
                       delta = clustering$delta)

df_clust %>% ggplot(aes(rho, delta)) +
  geom_point() +
  geom_hline(yintercept = 0.1, linetype = "dashed", color = "red") +
  geom_vline(xintercept = 30, linetype = "dashed", color = "red")
```

```{r}
# Silhouette of DPC
silh6 <- silhouette(clustering6$clusters, as.dist(mds_dis))
silh5 <- silhouette(clustering5$clusters, as.dist(mds_dis))
silh4 <- silhouette(clustering4$clusters, as.dist(mds_dis))
silh3 <- silhouette(clustering3$clusters, as.dist(mds_dis))
silh2 <- silhouette(clustering2$clusters, as.dist(mds_dis))

silh_plot <- c(mean(silh2[,3]),
```

```r
                mean(silh3[,3]),
                mean(silh4[,3]),
                mean(silh5[,3]),
                mean(silh6[,3]))

plot(c(2:6), silh_plot, type = "l", xlab = "Number of clusters", ylab = "Silhouette
    Coefficient")

plot(silh2, col = gray.colors(2), border = NA, main="Silhouette plot of k=2")
plot(silh3, col = gray.colors(3), border = NA, main="Silhouette plot of k=3")
plot(silh4, col = gray.colors(4), border = NA, main="Silhouette plot of k=4")
plot(silh5, col = gray.colors(5), border = NA, main="Silhouette plot of k=5")
plot(silh6, col = gray.colors(6), border = NA, main="Silhouette plot of k=6")
```

```{r}
# plotting the DPC result in different nmMDS projections
mds_plots <- mds[[4]][[1]] %>%
  data.frame() %>%
  cbind(as.factor(clustering4$clusters)) %>%
  set_colnames(c("pc1", "pc2", "pc3", "pc4", "cluster"))

mds_plots %>%
  ggplot(aes(x = pc2, y = pc1, color = cluster)) +
  geom_point(size = 0.1)
mds_plots %>%
  ggplot(aes(x = pc1, y = pc3, color = cluster)) +
  geom_point(size = 0.1)
mds_plots %>%
  ggplot(aes(x = pc1, y = pc4, color = cluster)) +
  geom_point(size = 0.1)
mds_plots %>%
  ggplot(aes(x = pc2, y = pc4, color = cluster)) +
  geom_point(size = 0.1)
```

```{r}
# start of code related to CTD embedding

knn <- nng(dx = dis_matrix, k = 5, mutual = FALSE, method = NULL,
    use.fnn = FALSE, algorithm = 'brute')

knn_matrix <- matrix(0, nrow=nrow(mds_dis), ncol=ncol(mds_dis))

for (i in 1:nrow(mds_dis)) {
  neigh = neighbors(knn, i, mode="out")
  neigh = as.numeric(neigh)
  knn_matrix[i, neigh] <- 1
}

set.seed(1997)
index1 <- sample(seq(1,nrow(mds_dis)), 50)
index2 <- sample(seq(1,nrow(mds_dis)), 50)
knn_matrix[index1,index2] <- 1

knn_matrix <- knn_matrix + t(knn_matrix)
```

```
knn_matrix[knn_matrix > 1] <- 1

knn_with_dist <- knn_matrix * mds_dis
sigma <- (sum(knn_with_dist)/sum(knn_matrix))
W <- exp(-(knn_with_dist)^2/(2*(sigma^2)))
W[W == 1] <- 0
D <- diag(rowSums(W))
volV <- sum(D)

# unnormalized Laplacian L
L <- D - W
eig <- eigen(L)

# normalized Laplacian L_sym
D_invsqrt <- inv(sqrt(D))
L_sym <- D_invsqrt %*% L %*% D_invsqrt
eig_sym <- eigen(L_sym)
```

```{r}
n <- length(eig$values)

# unnormalized L nwe coordinates in prep for CTD embedding
unnormal_coords <- map_dfc(seq(n-1, 1), function(.x) {
  tibble(
     sqrt(volV / eig$values[.x]) * eig$vectors[, .x]
  ) %>%
  set_colnames(.x)
}) %>% as.matrix()

# normalized L_sym new coordinates in prep for CTD embedding
normal_coords <- map_dfc(seq(n-1, 1), function(.x) {
  tibble(
     sqrt(volV / eig_sym$values[.x]) * eig_sym$vectors[, .x]
  ) %>%
  set_colnames(.x)
}) %>% as.matrix()

# dividing each row by inv square root of degree d_i
normal_coords <- D_invsqrt %*% normal_coords
```

```{r}
# unnormalized CTD euclidean distances
CTD_un_dis <- dist(unnormal_coords) %>% as.matrix()

# normalized CTD euclidean distances
CTD_n_dis <- dist(normal_coords) %>% as.matrix()
```

```{r}
cmMDS_unCTD <- cmdscale(CTD_un_dis, k=4) %>%
  data.frame() %>%
  set_colnames(c("pc1", "pc2", "pc3", "pc4"))

cmMDS_nCTD <- cmdscale(CTD_n_dis, k=4) %>%
  data.frame() %>%
  set_colnames(c("pc1", "pc2", "pc3", "pc4"))
```

```
```

```{r}
save(CTD_un_dis, file = "CTD_un_dis.Rda")
save(CTD_n_dis, file = "CTD_n_dis.Rda")
```

```{r}
cmMDS_unCTD %>%
  ggplot(aes(x = pc3, y = pc4)) +
  geom_point()
```

```{r}
plot(eig_sym$values[1480:1500])
```
```
```

References

Gower, John C (1971). "A general coefficient of similarity and some of its properties". In: *Biometrics*, pp. 857–871.

Lee, John A, Michel Verleysen, et al. (2007). *Nonlinear dimensionality reduction.* Vol. 1. Springer.

Hastie, Trevor et al. (2009). *The elements of statistical learning: data mining, inference, and prediction.* Vol. 2. Springer.

Rodriguez, Alex and Alessandro Laio (2014). "Clustering by fast search and find of density peaks". In: *science* 344.6191, pp. 1492–1496.

Watson, Peter (2014). URL: `https://imaging.mrc-cbu.cam.ac.uk/statswiki/FAQ/mds/stress`.

Wehrens, Ron and Johannes Kruisselbrink (2023). *Supervised and Unsupervised Self-Organising Maps.* URL: `https://cran.r-project.org/web/packages/kohonen/kohonen.pdf`.

Doe, Jane (2024). URL: `https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/hclust`.

Kraemer, Guido (2024). *coRanking.* URL: `https://cran.r-project.org/web/packages/coRanking/coRanking.pdf`.

Li, Chun-Biu (2024). *Lecture notes from course in unsupervised learning.*

Pedersen, Thomas Lin, Sean Hughes, and Xiaojie Qiu (2024). *Clustering by Fast Search and Find of Density Peaks.* URL: `https://cran.r-project.org/web/packages/densityClust/densityClust.pdf`.

Ripley, Brian (2024). *MASS package.* URL: `https://cran.r-project.org/web/packages/MASS/MASS.pdf`.