Assessing Behaviour of Casino Patrons Using Clustering Methods

by

Samuel Morrissette

A Thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfilment of the requirements of the degree of

Master of Science

Department of Statistics
University of Manitoba
Winnipeg

Abstract

Research into statistical clustering techniques has grown tremendously in the past several years due to advances in both theory and computational capability. Despite this growth, there has been little documented research pertaining to clustering within the context of the casino industry - likely due to the proprietary nature of data. However, clustering results can help identify structure within a given dataset and provide valuable information to stakeholders. In particular, the segmentation of casino patrons may allow casino operators and researchers to develop insight into gambling behaviour and tendencies. Consequently, it is a worthwhile endeavour to explore the application of clustering methods to casino data. First, we discuss in detail several clustering algorithms along with a variety of metrics to assess clustering validity. Next, we apply these algorithms to casino data provided by a local industry partner and compare the resulting partitions. Furthermore, we examine strategies for interpreting these clustering results. Finally, we propose different candidate models which have satisfactory statistical performance and result in meaningful clusters from a pragmatic perspective.

Acknowledgment Page

I would like to express my gratitude towards Dr. Saman Muthukumarana, whose great patience, support (both financial and moral), and guidance was unwavering throughout the thesis process.

I would also like to thank both Dr. Jason Fiege and Dr. Stasi Baran for their expertise, excellent suggestions, kindness, and humour that was always able to ease any stress I experienced.

I am thankful to the committee members, Dr. Pingzhao Hu and Dr. Po Yang, who took time out of their schedules to help me review and complete my thesis.

My heartfelt appreciation goes out to my family - Patrick, Debbie, Matthew, and Alana, who have always given me their love and support.

Finally, I would like to thank my partner Reagan, who has always been there for me and given me support throughout this whole journey.

Dedication Page

This work is dedicated to the memory of my mother, who always encouraged me to follow my dreams and believed in me every step of the way.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The casino industry is a multi-billion dollar global enterprise. Millions of people each year flock to casinos across the world for entertainment and the opportunity to gamble. As of 2019 in the U.S. alone, there were almost 1000 casino gaming locations employing 1.8 million people, and the indsutry was valued at an overall 261 billion USD (American Gaming Association, 2019). As a result, it is crucial for casino operators to invest time and resources into creating an enjoyable experience for their customers. Such experiences will encourage retention of existing customers in addition to attracting new ones. It is necessary, then, that casino operators are able to understand and cater towards to their customers.

One of the most popular forms of gambling within casinos is the slot machine. Unsurprisingly, for the casinos themselves, slot machines happen to be one of the most profitable activities. For the average Las Vegas casino in 2017, it was found that 50.5% of gaming revenue was from slot machines alone (Schwartz, 2017). In some states, this number is estimated to be even higher; between 65

and 80 percent (Schwartz, 2018). It should be evident, then, that slot machines are an activity in which patrons choose to spend a large portion of their time and wallet. Consequently, it is important that casino operators analyze slot machine usage to develop valuable insight into their player base.

1.1 Motivation

We obtained a casino dataset from a local industry partner with seven months of slot machine activity for several thousand casino patrons. Within the casino, there were over 400 machines active during this time. The dataset included variables such as date and time of play, unique machine identifiers, duration of slot machine play, and the number of spins a patron played on the machine. Furthermore, the dataset excluded all demographic information such as gender and age. Despite this absent information, we wish to analyze the behaviour of casino patrons based on the present variables. Ideally, this analysis should result in actionable conclusions for casino operators.

A common method of analyzing customer behaviour within a variety of industries is known as "clustering". Statistical clustering, or just clustering, is the process by which similar objects are placed together into a group (called a cluster), while simultaneously placing differing objects into separate groups. By partitioning the dataset into distinct clusters, it is possible to discover structure within the data that may have important, real-world applications. For instance, clustering has been used to study customer behaviour within

industries such as online retail (Fry and Manna, 2016), credit card, (Wu and Lin, 2005), banking (Farajian and Mohammadi, 2010), and several others. By using clustering methods, stakeholders within these industries may be able to learn new information about their customers and apply this knowledge through wise marketing strategies to ultimately increase profitability. In particular, the application of clustering methods to casino patrons based on their slot machine usage can help operators and researchers answer a variety of questions. For example:

- How are customers spending their time?
- Where are customers spending their money?
- What changes can be made within the casino to cater towards customer needs?

Clustering is a form of unsupervised learning, which means that there is an absence of ground truth. In other words, we are unsure of which customer(s) belongs to a given cluster, or if any clustering even exists in the first place. This is in contrast to supervised learning, in which there is ground truth present and we are training an algorithm to classify data or predict an outcome (Delua, 2021). Thus, clustering is often a complicated task that requires careful attention.

There are several clustering algorithms available which result in different clustering outputs and therefore different interpretations. We present, explore, assess, and contrast several of these algorithms in detail. Since each algorithm results in a different output, it is of great importance to present metrics and

techniques to distinguish among "good" and "bad" clustering results. Without any ground truth, determining the validity of outputs can be challenging. Therefore, we also present and discuss several evaluation metrics that can be used to assess clustering outputs.

Although some research has been conducted in clustering casino patrons (Iaci and Singh, 2012), results are difficult to reproduce due to the confidentiality of casino data. Instead, most clustering research tend to focus on the motivation of gamblers instead of actual gambling behaviour within the casino (Phillips et al., 2004) (Lee et al., 2006) (Chen et al., 2013). These studies are often conducted using questionnaires or surveys rather than data obtained from the casino itself. Without this demographic information present in our dataset, clustering casino patrons becomes a different challenge. Nevertheless, such an endeavour may prove worthwhile for operators. In the context of slot machine gambling, for example, stakeholders may gain beneficial information into which patrons are attracted to certain slot machines and why that may be.

Using the aforementioned casino dataset, we apply various clustering algorithms and techniques. In particular, we cluster casino patrons in two different ways. Firstly, we cluster patrons based solely on their slot machine activity using variables such as their frequency and duration of play. To accomplish this, we use continuous covariates within the cluster analysis. Secondly, we cluster casino patrons based on their slot machine preference. That is, we determine if certain patrons gravitate more towards certain machines. For this cluster analysis, we use binary covariates (covariates that can only take on

two distinct values). Furthermore, we analyze the results of both clustering methods using evaluation metrics and present visualizations that aid in understanding the output partitions. Finally, we give meaning to the resulting clusters by discussing possible interpretations. This last step is likely the most important step for stakeholders who may eventually utilize this information to make evidence-based business decisions.

1.2 Thesis Overview

In Chapter 2, we present in detail several clustering algorithms that can be used on a wide variety of data. In Chapter 3, we present various clustering evaluation metrics that are used to assess and validate clustering outputs from the algorithms in the previous chapter. Chapter 4 pertains to the application of these algorithms, techniques, and evaluation metrics to real casino data. Furthermore, we discuss a strategy to interpret clustering output in order to ultimately provide value to stakeholders. Finally, in Chapter 5 we conclude the thesis with a brief discussion of the results and potential topics for future research.

Chapter 2

Clustering Methodology

2.1 Principal Components Analysis

Principal Components Analysis (PCA) (F.R.S., 1901) is a technique developed in the early 1900's used to reduce the dimension of a dataset while maximizing the amount of information retained in doing so. The purpose of dimension reduction, in this context, may be to increase the interpretability of analyses by focusing on fewer variables and to enable visualization of the dataset by reducing it to 2 or 3 dimensions. PCA reduces the dimension of a dataset and minimizes information loss by discovering variables that are uncorrelated with one another and are linear combinations of the original variables.

Suppose we have a dataset $\mathbf{X}$ with n observations on p variables (i.e. the dimension of the dataset is p). $\mathbf{X}$ may be written as an $n \times p$ matrix,

$$\mathbf{X} = \begin{pmatrix} \mathbf{x_1} & \mathbf{x_2} & \dots & \mathbf{x_p} \end{pmatrix}.$$

Our objective is to find a linear combination of these original variables that

maximizes the variance. Given a $p \times 1$ vector of constants, $\mathbf{a} = (a_1, a_2, \ldots, a_p)^T$, linear combinations of $\mathbf{x_i}$ are of the form

$$\sum_{i=1}^{p} a_i \mathbf{x_i} = \mathbf{Xa}$$

The variance of this linear combination is

$$\text{Var}(\mathbf{Xa}) = \mathbf{a}^T \text{Var}(\mathbf{X})\mathbf{a}.$$

However, in order to ensure a unique solution, there is an additional constraint that the vector $\mathbf{a}$ must have unit norm. That is,

$$\|\mathbf{a}\| = \mathbf{a}^T\mathbf{a} = 1.$$

Therefore, to recapitulate, our overall objective is to find an $\mathbf{a}$ that maximizes $\mathbf{a}^T\text{Var}(\mathbf{X})\mathbf{a}$, under the constraint that $\mathbf{a}^T\mathbf{a} = 1$.

Differentiation, in conjunction with Lagrange multipliers, is applied to the above to obtain the following equation:

$$\text{Var}(\mathbf{X})\mathbf{a} = \lambda\mathbf{a},$$

where λ is a scalar quantity. Therefore, it is clear that $\mathbf{a}$ is an eigenvector of $\text{Var}(\mathbf{X})$, and λ is a corresponding eigenvalue. Note that

$$\begin{aligned} \text{Var}(\mathbf{Xa}) &= \mathbf{a}^T\text{Var}(\mathbf{X})\mathbf{a} \\ &= \mathbf{a}^T\lambda\mathbf{a} \\ &= \lambda\mathbf{a}^T\mathbf{a} \end{aligned}$$

$$= \lambda \qquad \text{since } \mathbf{a}^T\mathbf{a} = 1.$$

So, it is clear that we must choose the largest eigenvalue of $\text{Var}(\mathbf{X})$ to maximize $\text{Var}(\mathbf{Xa})$. The linear combination, $\mathbf{Xa}$, is known as a principal component (PC). Furthermore, the elements of the vector $\mathbf{a}$ are known as the PC loadings, and the elements of $\mathbf{Xa}$ are known as the PC scores.

Consider two distinct PC's, $\mathbf{Xa}_i$ and $\mathbf{Xa}_j$, with $i \neq j$. Then, the covariance between them is

$$\begin{aligned}\text{Cov}\left(\mathbf{Xa}_i, \mathbf{Xa}_j\right) &= \mathbf{a}_j^T \text{Var}\left(\mathbf{X}\right)\mathbf{a}_i \\ &= \mathbf{a}_j^T \lambda_i \mathbf{a}_i \\ &= \lambda_i \mathbf{a}_j^T \mathbf{a}_i,\end{aligned}$$

which is 0 if $\mathbf{a}_i^T\mathbf{a}_j^T = 0$. Moreover, the correlation is 0 (i.e. the two PC's are uncorrelated) if this is the case, since

$$\text{Cor}\left(\mathbf{Xa}_i, \mathbf{Xa}_j\right) = \frac{\text{Cov}\left(\mathbf{Xa}_i, \mathbf{Xa}_j\right)}{\sqrt{\text{Var}\left(\mathbf{Xa}_i\right)\text{Var}\left(\mathbf{Xa}_j\right)}}.$$

Therefore, to find subsequent PC's, we can proceed as before, but with the additional constraint that $\mathbf{a}_i^T\mathbf{a}_j^T = 0$ for all $i \neq j$ to ensure that distinct PC's are uncorrelated. The result is that subsequent PC's are calculated by finding the subsequent eigenvectors of $\text{Var}\left(\mathbf{X}\right)$. So, for example, to find the second PC, we would find the eigenvector of $\text{Var}\left(\mathbf{X}\right)$ corresponding to the second largest eigenvalue. In this manner, we can calculate up to p principal components.

Of course, since the variance of $\mathbf{X}$ is usually unknown, in practice we must estimate the PC loadings using the eigenvalues of the sample covariance matrix, $\mathbf{S}$, which is defined as

$$\mathbf{S} = \frac{1}{n-1} \left(\mathbf{X} - \mathbf{1}_n \bar{\mathbf{x}}\right)^T \left(\mathbf{X} - \mathbf{1}_n \bar{\mathbf{x}}\right),$$

where $\bar{\mathbf{x}}$ is a $1 \times p$ mean vector and $\mathbf{1}_n$ is a $n \times 1$ vector of 1's. Note that the resulting matrix, $\mathbf{S}$, is of dimension $p \times p$.

One final (and important) note is that because the variance matrix of $\mathbf{X}$ is dependent upon the scale of the variables, the resulting PC's may be greatly affected if the scales of variables differ drastically. Thus, it is common to standardize the variables (by subtracting their mean and dividing by their standard deviation) prior to calculating any PC's.

2.1.1 Dimension-Reduction using PCA

Recall that our dataset is of dimension p. To reduce the dimension from p to k with $k < p$, PCA is often employed and the first k components are selected. The k components are often selected based on the *proportion of explained variance* by the components. Recall that the i^{th} principal component has variance λ_i. Then, the following equations hold true:

$$\text{Total Variance} = \sum_{i=1}^{p} \lambda_i$$

$$\text{Variance Explained by } k \text{ components} = \sum_{i=1}^{k} \lambda_i$$

$$\text{Proportion of Explained Variance} = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{p} \lambda_i}.$$

A technique used to visualize the proportion of explained variance is a scree plot (Cattell, 1966). A scree plot shows the principal components along the horizontal axis, along with their respective proportion of explained variance on the vertical axis.

By inspecting the scree plot and identifying the start of any "elbows" in the plot, one can determine a reasonable number of components to select. The elbow represents the point at which there are diminishing returns in terms of increasing the number of PC's to increase the proportion of explained variance.

2.2 K-Means

K-means clustering (Macqueen, 1967) is one of the most popular clustering methods. The algorithm works by partitioning a dataset into a pre-specified number of non-overlapping clusters. That is, each data point belongs to one (and only one) of the clusters resulting from the algorithm.

The k-means problem minimizes the within-cluster variation of each cluster. That is, let $C_1, C_2, \ldots, C_k$ represent the dataset partitioned into k clusters.

Then, the k-means algorithm will choose $C_1, C_2, \ldots, C_k$ in the following manner:

$$\min_{C_1,C_2,\ldots,C_k} \left(\sum_{i=1}^{k} W(C_i) \right),$$

where $W(C_i)$ is the within-cluster variation of cluster i, defined as

$$W(C_i) = \sum_{x^{(j)} \in C_i} \left\| x^{(j)} - \mu_i \right\|^2,$$

where μ_i denotes the center of cluster i and $\|\cdot\|$ denotes the Euclidean norm. That is, the within-cluster variation defined above is the sum of squared Euclidean distances between each data point in a given cluster and that cluster's center.

Therefore, using the above, we obtain the following objective function:

$$\min_{C_1,C_2,\ldots,C_k} \left(\sum_{i=1}^{k} \sum_{x^{(j)} \in C_i} \left\| x^{(j)} - \mu_i \right\|^2 \right).$$

Then, the objective of k-means is two-fold: choose the center of each cluster so that it is as close as possible to the data points assigned to that cluster, and assign the data points that are close to a center to that cluster.

There are several algorithms that implement k-means clustering, one of which is Lloyd's algorithm (Lloyd, 1982). First, Lloyd's algorithm begins by randomly choosing k different initial cluster centers. Then, each data point is assigned to the cluster with the nearest center. The centers are then updated

using the mean of all points belonging to that cluster. The steps of assigning data points to the nearest cluster and then recalculating the centers reoccurs until some stopping criterion is achieved. Lloyd's algorithm is given in full in algorithm 1.

Algorithm 1: Lloyd's Algorithm

1 Generate k initial cluster centers, denoted μ_i, with $1 \leq i \leq k$

2 For each data point x, calculate:

$$\|x - \mu_i\|^2, \quad 1 \leq i \leq k$$

and assign x to the cluster with the smallest distance to the center.

3 Recalculate the centers as the mean of the data points currently assigned to that cluster:

$$\mu_i = \frac{1}{n_i} \sum_{x^{(i)} \in C_i} x^{(i)},$$

where n_i is the number of data points currently in cluster C_i.

4 If there has been no reassignment of data points, then end the algorithm. otherwise, go to step 2.

Note that Lloyd's algorithm is sensitive to the choice of initial cluster centers in the first step. Therefore, a wise choice of initial centers is vital to the efficient convergence of the algorithm. There are several ways of accomplishing this task, such as using previous knowledge, assigning k random observations in the dataset as initial centers, or choosing k random points in the appropriate dimension.

In the statistical software, R (R Core Team, 2020), the *kmeans* function provides an "nstart" argument that allows the user to specify a number of starts to the algorithm. The function will then return the clustering partition

with the smallest within-cluster variance based on these starts.

2.3 K-medoids

K-medoids (Kaufman and Rousseeuw, 1990) partitions a dataset into a pre-specified number of non-overlapping clusters, similar to k-means. However, unlike k-means, k-medoids uses actual data points in the dataset as the cluster centers, known as the cluster medoids. Using cluster medoids can greatly reduce the effect of noise and outliers on cluster outputs. Another advantage of using k-medoids is that the within-cluster variation can be defined using metrics other than the Euclidean distance, such as the Manhattan distance (Krause, 1973). As a result, k-medoids clustering allows for greater flexibility compared to k-means clustering.

Like the k-means problem, the k-medoids problem minimizes the within-cluster variation of each cluster:

$$\min_{C_1,C_2,\ldots,C_k} \left(\sum_{i=1}^{k} W(C_i) \right),$$

where $W(C_i)$ once again denotes the within-cluster variation of cluster i. As mentioned above, the within-cluster variation can be defined using a variety of distance metrics. In general, let this distance be denoted d. Then, the k-medoids objective function becomes

$$\min_{C_1,C_2,\ldots,C_k} \left(\sum_{i=1}^{k} \sum_{x^{(j)} \in C_i} d(x^{(j)}, y_i) \right),$$

where y_i denotes the medoid of cluster i. Note that each medoid is a member of the dataset. That is, if we let $\mathcal{X}$ represent the full dataset, then

$$y_1, y_2, \ldots, y_k \in \mathcal{X}.$$

The most common algorithm for finding a solution to the k-medoids problem is known as the Partitioning Around Medoids algorithm (PAM) (Kaufman and Rousseeuw, 1990). PAM is implemented in R within the "cluster" library (Maechler et al., 2019).

2.4 Hierarchical Clustering

Hierarchical clustering (Kaufman and Rousseeuw, 1990) is a form of clustering in which a certain hierarchy of clusters are created. In particular, clusters are nested within each other. In the following sections, we will explore the two types of hierarchical clustering: agglomerative and divisive.

Unlike k-means, hierarchical clustering does not need to be initialized with a choice of the number of clusters. Instead, all choices of the number of clusters, from 1 to n, where n is the number of data points in the dataset, are created by the hierarchical clustering algorithms. Obviously, it is still necessary to actually choose an appropriate number of clusters for the data after the algorithm has completed. The output of hierarchical clustering is often viewed in the form of a "dendrogram" and may be used to help inform the choice of number of clusters. Dendrograms will be discussed in further detail.

It should be noted that, unlike k-means, hierarchical clustering does not

depend on any kind of random initialization. Therefore, the result of hierarchical clustering are reproducible. On the other hand, hierarchical clustering is more computationally expensive with large datasets.

2.4.1 Agglomerative Hierarchical Clustering

In agglomerative clustering (Kaufman and Rousseeuw, 1990), each observation in the dataset begins in its very own cluster (i.e. n distinct clusters). As the algorithm runs, clusters are joined together until, eventually, all clusters have joined together (i.e. one single cluster remains).

With the above in mind, the obvious question is how to appropriately join the clusters together. In each step of the algorithm, the two most similar clusters are joined together to create one cluster. However, "most similar" is a somewhat more complicated term than it may initially appear to be, especially since we are now dealing with groups of observations. In fact, there are numerous options to define similarity in the context of hierarchical clustering.

Similarity is defined using a combination of a particular distance metric and linkage criterion. The linkage criterion is how the (dis)similarity between groups of observations is defined. For example, we could calculate the pairwise distances between all observations in two distinct groups and then take the maximum of those distances. This is known as *complete-linkage*. We can also use *single-linkage*, which is taking the minimum of all pairwise distances in two distinct groups instead of the maximum. Or, we can use *centroid-linkage*, whereby we calculate the centroid of both groups and then simply use the

distance between these centroids. Another popular choice of linkage is the *average-linkage*, which is using the mean of all pairwise distances between data points in the two groups. These linkage criteria are summarized in Table 2.1. Note that several other linkage criteria are available. Furthermore, when calculating a particular linkage criterion, a distance metric must be chosen. Potential distance metrics include the Euclidean distance, squared Euclidean distance, Manhattan distance, Mahalanobis distance, and more.

With a chosen linkage-criterion and distance metric, the agglomerative clustering algorithm proceeds as shown in algorithm 2. Agglomerative hierarchical clustering is implemented in R using the *agnes* function from the "cluster" library (Maechler et al., 2019).

Linkage Criterion	Formula
Complete-linkage	$\max d(x_a, x_b) \ : x_a \in C_i, x_b \in C_j$
Single-linkage	$\min d(x_a, x_b) \ : x_a \in C_i, x_b \in C_j$
Centroid-linkage	$d(c_i, c_j)$, where c_i is the center of cluster C_i and c_j is the center of cluster C_j.
Average-linkage	$\frac{1}{n_i n_j} \sum_{x_a \in C_i} \sum_{x_b \in C_j} d(x_a, x_b)$, where n_i and n_j are the respective number of data points in cluster i and cluster j.

Table 2.1: Linkage criteria for agglomerative hierarchical clustering

2.4.2 Divisive Hierarchical Clustering

Divisive hierarchical clustering (Kaufman and Rousseeuw, 1990) is simply the inverse of agglomerative clustering. Here, all observations start in one single

Algorithm 2: Agglomerative Hierarchical Clustering

1 Begin the algorithm with each observation in its own cluster (i.e. n distinct clusters).
2 Calculate the dissimilarity, using the chosen linkage criterion and distance metric, between each cluster.
3 Fuse the two least dissimilar clusters (i.e. the most similar clusters).
4 Go to step 2 and continue iterating until only one cluster remains.

cluster. Each step of the algorithm proceeds by splitting a cluster into two distinct clusters until n clusters remain. However, it is evident that when a dataset contains n points, the first step of the algorithm (splitting the dataset into two distinct clusters) must consider $2^{n-1} - 1$ ways of splitting the data. Thus, even in relatively small datasets, this computation is infeasible. Instead of considering all possible ways of splitting the data, a smarter algorithm that only considers a subset of possibilities is employed.

In the first step of the algorithm, a dissimilarity matrix is calculated between all observations of the dataset. Note that this dissimilarity matrix may be calculated using a variety of distance metrics. The most dissimilar observation then creates a "splinter" cluster. Then, the dissimilarity matrix of the original cluster (minus the splintered observation) is recalculated, along with each observation's dissimilarity to the one splintered observation. If any observations in the original clusters are more similar to the splintered cluster than to its current cluster, then the most dissimilar observation is moved to the splintered group. Once again, the dissimilarity matrix is recalculated for the two clusters. The movement between observations of the original cluster to

the splintered cluster and recalculation of the dissimilarity matrix continues until all observations of the original cluster are more similar to each other than to the splintered cluster. This completes step one of the algorithm.

Now that we have two cluster (i.e. the original and the splintered cluster), we have to decide which of these to divide. This is decided by finding the cluster with the largest dissimilarity between any two of its observations. Once again, the most dissimilar observation in that cluster forms the new splinter group. As in step one, dissimilarity matrices are calculated for each cluster and compared to the dissimilarity of observations in the original clusters to the splinter observation. The most dissimilar observation is then moved to the splinter group and the algorithm proceeds. These steps are continued until each observation is in its own cluster.

The algorithm discussed above is summarized in algorithm 3. Furthermore, divisive hierarchical clustering is implemented through the *diana* function from the cluster library Maechler et al. (2019) in R.

2.4.3 Dendrograms

A dendrogram is a way of visually representing hierarchical clustering. An example is shown in Figure 2.1. At the bottom of the dendrogram, each observation in the dataset is a single "leaf". As we move up the tree, leafs join together based on their similarity to each other and create branches. That is, leafs that are joined together closer to the bottom of the dendrogram are more similar to each other than leafs that are joined to a branch further up the tree.

Algorithm 3: Divisive Hierarchical Clustering

1 Begin the algorithm with all observations in a single cluster.
2 If this is the first iteration of the algorithm, calculate the dissimilarity matrix and move the most dissimilar observation to form a splinter cluster. Otherwise, find the cluster with the largest dissimilarity between any two of its observations and use the most dissimilar observation from this cluster to form a splinter cluster.
3 Recalculate dissimilarity matrices. If any observations are more similar to the splinter cluster than to its current cluster, move the most dissimilar observation to the splinter cluster.
4 Go to step 3 and repeat until all observations are more similar to their current cluster than to the splinter cluster.
5 Go to step 2 until all observations are in their own clusters (i.e. n clusters exist).

The vertical axis, then, is a measure of dissimilarity. By producing a horizontal "cut" across the dendrogram, we can therefore obtain our desired clusters. The branches that intersect our horizontal cut are the distinct clusters, and each observation/leaf belonging to that branch is in that particular cluster.

2.5 Fuzzy c-means

Up until this point, we have only considered clustering methods that partition the data into one cluster or the other with absolute certainty. That is, there has been no probability or uncertainty associated with the output. This is known as *hard clustering*. Conversely, in *fuzzy clustering* (also known as *soft clustering* (Dunn, 2008), each data point can belong to more than one partition with a certain probability. One such popular soft clustering technique is known

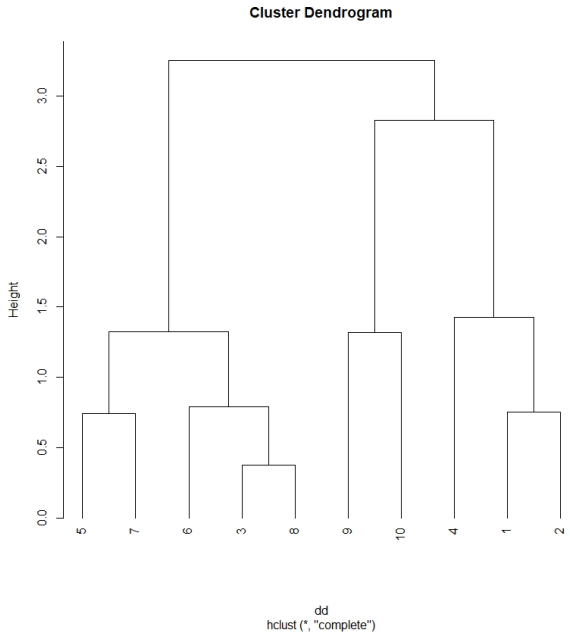

Figure 2.1: Example Dendrogram

as *fuzzy c-means*. The description of fuzzy c-means is given as follows:

Let $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ be our dataset that we wish to partition into c fuzzy clusters. Then, the following loss function, f, is used for the fuzzy c-means algorithm:

$$f\left[U, \mathcal{X}, V\right] = \sum_{i=1}^{n} \sum_{k=1}^{c} u_{ik}^{m} \left\|x_i - v_k\right\|^2,$$

where v_k is the centroid of cluster k, u_{ik} is the degree of membership of data point i to cluster k, and m is a parameter that determines the "fuzziness" of the clusters and is between $[1, \infty]$. The value of m is often defaulted to 2.

The degree of membership, u_{ik}, must satisfy the following conditions:

$$u_{ik} \in [0, 1] \quad \forall i, k$$

$$\sum_{k=1}^{c} u_{ik} = 1 \quad \forall i$$

$$\sum_{i=1}^{n} u_{ik} > 0 \quad \forall k.$$

Then, in order to minimize the objective function, f, with the constraints above, the following formulas for u_{ik} and v_k are used in the fuzzy c-means algorithm (Bezdek, 1981):

$$u_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{\|\mathbf{x}_i - \mathbf{v}_k\|}{\|\mathbf{x}_i - \mathbf{v}_j\|} \right)^{\frac{2}{m-1}}}$$

$$v_k = \frac{\sum_{i=1}^{n} \left(u_{ik}\right)^m x_i}{\sum_{i=1}^{n} \left(u_{ik}\right)^m}.$$

The full algorithm for fuzzy c-means is given in algorithm 4. The fuzzy c-means algorithm is implemented in R under the "ppclust" package (Cebeci, 2019).

Algorithm 4: Fuzzy c-means Algorithm

1 Given a pre-specified number of clusters c and dataset $\mathcal{X} = \{x_1, \ldots, x_n\}$, choose c random cluster centers, $v_1, v_2, \ldots, v_c$.

2 Calculate the degree of membership:

$$u_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{\|\mathbf{x}_i - \mathbf{v}_k\|}{\|\mathbf{x}_i - \mathbf{v}_j\|} \right)^{\frac{2}{m-1}}}, \quad \text{for } 1 \leq i \leq n \text{ and } 1 \leq k \leq c$$

3 Compute the new cluster centers:

$$v_k = \frac{\sum_{i=1}^{n} (u_{ik})^m x_i}{\sum_{i=1}^{n} (u_{ik})^m}, \quad \text{for } 1 \leq k \leq c$$

4 If $\max \left| u_{ik}^{(t+1)} - u_{ik}^{(t)} \right| < \epsilon$, where ϵ is a small positive number chosen as a stopping criterion, and t tracks the number of iterations, then the algorithm stops. Otherwise, return to step 2.

2.6 Model-based Clustering

Model-based, or distribution-based, clustering is a method in which it is assumed the data originate from some mixture of probability distributions. Unlike other techniques such as k-means or hierarchical clustering which are heuristic in nature, model-based clustering relies on an underlying model. In this type of clustering, each component of the mixture model corresponds to a cluster. Similar to the fuzzy c-means algorithm, model-based clustering partitions the dataset using soft clustering. An observation's assignment to a

particular cluster is the probability that the observation originated from that component. An advantage to using model-based clustering is that we are able to use the Bayesian information criterion (BIC) (Schwarz, 1978) and/or Akaike information criterion (AIC) (Akaike, 1974) for model selection.

Concretely, suppose that we have a dataset of n observations, $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$. The data is assumed to have been generated from an underlying mixture distribution (McLachlan and Peel, 2000):

$$f(x_i|\Theta) = \sum_{j=1}^{k} \pi_j f_j\left(x_i|\theta_j\right).$$

The variables in the equation above are as follows:

- π_j is the mixing weight of the j^{th} component with

$$\sum_{j=1}^{k} \pi_j = 1$$

$$\pi_j > 0 \quad \text{for all } j$$

- f_j is the probability density function of the j^{th} component.
- θ_j are the parameter(s) of the j^{th} probability density function
- Θ is the full set of model parameters. That is, $\Theta = \{\pi_1, \ldots, \pi_k, \theta_1, \ldots, \theta_k\}$.

2.6.1 The Gaussian Mixture Model

The most popular form of model-based clustering is Gaussian mixture models (GMM), where the underlying model is a mixture of Gaussian distributions.

Given a p-dimensional dataset and a number of clusters, k, the goal of the model-based clustering algorithm is to fit a Gaussian mixture distribution with k components to the dataset.

The Gaussian Mixture distribution can be written as (Bishop, 2006)

$$f(x_i|\Theta) = \sum_{j=1}^{k} \pi_j \, \phi\left(x_i|\mu_j, \Sigma_j\right),$$

where $\phi\left(x_i|\mu_j, \Sigma_j\right)$ denotes the multivariate normal probability density function with p-dimensional mean vector μ_j and $p \times p$ covariance matrix Σ_j. This p.d.f. is defined as

$$\phi\left(x|\mu, \Sigma\right) = \frac{\exp\left[-\frac{1}{2}\left(x-\mu\right)^T \Sigma^{-1}\left(x-\mu\right)\right]}{\sqrt{(2\pi)^p|\Sigma|}}.$$

The multivariate normal distribution is ellipsoidal and centered at the the mean-vector μ. The covariance parameter, Σ, controls aspects of the geometry of the distribution such as its volume, shape, and orientation (Scrucca et al., 2016a). Therefore, by imposing constraints upon the covariance matrices of the mixture components, we can obtain a number of different clustering results. We will examine some of these constraints in a later section and investigate the resulting clustering output.

To fit a mixture of Gaussians to the dataset, we will proceed by examining the log-likelihood of the distribution (Bishop, 2006). Note that we have assumed that each x_i is independent and identically distributed. Therefore, the likelihood

can be written as

$$L(\Theta|x_1, \ldots, x_n) = \prod_{i=1}^{n} f(x_i|\Theta).$$

To simplify, we take the logarithm of both sides to obtain the log-likelihood, denoted ℓ:

$$\begin{aligned}\ell(\Theta|x_1, \ldots, x_n) &= \sum_{i=1}^{n} \log f(x_i|\Theta) \\ &= \sum_{i=1}^{n} \log \left[\sum_{j=1}^{k} \pi_j \, \phi\left(x_i|\mu_j, \Sigma_j\right) \right].\end{aligned}$$

Recall that $\Theta = \{\pi_1, \ldots, \pi_k, \theta_1, \ldots, \theta_k\}$. Our goal is to find the value of Θ that maximizes the log-likelihood, given our observed data. However, maximizing the log-likelihood using analytical methods is often difficult or impossible. As such, we can simplify the task by introducing a *latent* variable.

A latent variable is a variable that is not directly observed and thus has to be inferred in some way from the data. Therefore, latent variables are also known as hidden variables. For instance, we will introduce a latent variable to our mixture model that indicates from which component a particular data point was drawn.

Let this latent variable be denoted $z_i = \{z_{i1}, z_{i2}, \ldots, z_{ik}\}$, where z_{ij} takes on the value 0 if x_i was not drawn from component j, and 1 if x_i was drawn from component j. Then, we can write the following:

$$z_i \sim \text{Multinomial}(1, \boldsymbol{\pi}) \quad \text{where } \boldsymbol{\pi} = \{\pi_1, \ldots, \pi_k\}$$

$$f(x_i|z_i,\Theta) = \prod_{j=1}^{k} f_j(x_i|\theta_j)^{z_{ij}}.$$

Suppose that we could observe the z_i's, this would result in

$$\begin{aligned} f(x_i, z_i|\Theta) &= f(x_i|z_i,\Theta)f(z_i|\Theta) \\ &= \prod_{j=1}^{k} f_j(x_i|\theta_j)^{z_{ij}} \prod_{j=1}^{k} \pi_j^{z_{ij}} \quad \text{Since } z_i \text{ is multinomial distributed} \\ &= \prod_{j=1}^{k} (\pi_j f_j(x_i|\theta_j))^{z_{ij}} . \end{aligned}$$

Therefore, we can proceed with deriving the log-likelihood of the above density. In this instance, the log-likelihood is known as the "complete-data log-likelihood".

$$\begin{aligned} L(\Theta|z_1,\ldots,z_n,x_1,\ldots,x_n) &= \prod_{i=1}^{n} f(x_i, z_i|\Theta) \\ &= \prod_{i=1}^{n}\prod_{j=1}^{k} (\pi_j f_j(x_i|\theta_j))^{z_{ij}} \\ \ell(\Theta|z_1,\ldots,z_n,x_1,\ldots,x_n) &= \sum_{i=1}^{n}\sum_{j=1}^{k} z_{ij} \log\Big[\pi_j f_j(x_i,|\theta_j)\Big]. \end{aligned}$$

Now, it is much easier to derive maximum likelihood estimators for the parameters π_j, μ_j, and Σ_j with the complete-data log-likelihood. This is done in the usual way (taking the derivative with respect to each variable and setting it

equal to 0), although π_j has a constraint of $\sum_{j=1}^{k} \pi_j = 1$ and so a Lagrange multiplier must be used. The end results are:

$$\pi_j^* = \frac{1}{n} \sum_{i=1}^{n} z_{ij}$$

$$\mu_j^* = \frac{\sum_{i=1}^{n} z_{ij} x_i}{\sum_{i=1}^{n} z_{ij}}$$

$$\Sigma_j^* = \frac{\sum_{i=1}^{n} z_{ij} (x_i - \mu_j^*)(x_i - \mu_j^*)^T}{\sum_{i=1}^{n} z_{ij}}.$$

So, if we knew all of $z_1, \ldots, z_n$, we would be able to easily fit a mixture of multivariate Gaussians using the maximum likelihood estimators above and infer which data points were drawn from which component with a certain degree of confidence. However, given that this is not the case, we have to turn to an iterative algorithm known as Expectation-Maximization.

2.6.2 Expectation-Maximization Algorithm

The Expectation-Maximization, or EM, algorithm (Dempster et al., 1977) is used to find maximum likelihood estimates for parameters when latent variables are present. First, we initialize all of π, μ, Σ. Then, we treat these parameters as fixed and compute a "guess" of z_{ij}. In other words, we will find estimates for all of z_{ij} given our current parameters (E-step). Then, we update these parameters using our estimates of z_{ij} from the previous step in combination with the maximum likelihood estimator formulas (M-step).

In the E-step, we calculate the probability that a data point belongs to each cluster (often referred to as the *responsibility*) using current values of the parameters:

$$\begin{aligned}\tau_{ij} &= f(z_i = j|x_i, \Theta) \\ &= \frac{f(x_i|z_i = j, \Theta) f(z_i = j|\Theta)}{f(x_i|\Theta)} \quad \text{by Bayes' Theorem} \\ &= \frac{\phi(x_i|\mu_j, \Sigma_j)\pi_j}{\sum_{l=1}^{k} \phi(x_i|\mu_l, \Sigma_l)\pi_l}.\end{aligned}$$

In the M-step, we update our parameters using τ_{ij}, calculated in the E-step:

$$\begin{aligned}\pi_j^{\mathbf{new}} &= \frac{1}{n}\sum_{i=1}^{n} \tau_{ij} \\ \mu_j^{\mathbf{new}} &= \frac{\sum_{i=1}^{n} \tau_{ij} x_i}{\sum_{i=1}^{n} \tau_{ij}} \\ \Sigma_j^{\mathbf{new}} &= \frac{\sum_{i=1}^{n} \tau_{ij}(x_i - \mu_j^{\mathbf{new}})(x_i - \mu_j^{\mathbf{new}})^T}{\sum_{i=1}^{n} \tau_{ij}}.\end{aligned}$$

Finally, the log-likelihood is computed using the new parameter values found in the M-step:

$$\ell(\Theta|x_1, \ldots, x_n) = \sum_{i=1}^{n} \log \left[\sum_{j=1}^{k} \pi_j \, \phi\left(x_i|\mu_j^{\mathbf{new}}, \Sigma_j^{\mathbf{new}}\right) \right].$$

If convergence of the log-likelihood has been reached according to some criterion, then the algorithm stops. If not, the E and M steps are repeated until it has been reached.

A full treatment of the EM algorithm for Gaussian mixture models can be found in (Bishop, 2006).

2.6.3 Constraints on Covariance in the Gaussian Mixture Model

Recall that the Gaussian Mixture model is written as

$$f(x_i|\Theta) = \sum_{j=1}^{k} \pi_j \, \phi\left(x_i|\mu_j, \Sigma_j\right).$$

where the covariance matrix, Σ_j, defines certain geometric properties of the j^{th} Gaussian component. Thus, by imposing some constraints upon the covariance matrix of each component, we may, in turn, control the shape of the resulting clusters.

The covariance matrix of the j^{th} component can be decomposed using eigendecomposition into

$$\Sigma_j = \lambda_j D_j A_j D_j^T,$$

where D_j is an orthogonal matrix of eigenvectors, A_j is a diagonal matrix with $\det(A) = 1$ and the elements on the diagonal are the normalized eigenvalues of Σ_j, and $\lambda_j = \det(\Sigma_J)^{\frac{1}{d}}$ (d is the dimension). With this decomposition, it can be shown that D_j controls the orientation of the ellipsoid, A_j determines the shape of the density contours, and λ_j determines the volume of the ellipsoid (Scrucca et al., 2016a). Some mixture models allow each component's covariance

matrix to vary, while others assume a fixed covariance. Table 2.2 shows the available models in the "mclust" (Scrucca et al., 2016b) package in R. Using the

Model	Σ_j	Distribution	Volume	Shape	Orientation
EII	λI	Spherical	Equal	Equal	–
VII	$\lambda_j I$	Spherical	Variable	Equal	–
EEI	λA	Diagonal	Equal	Equal	Coordinate Axes
VEI	$\lambda_j A$	Diagonal	Variable	Equal	Coordinate Axes
EVI	λA_j	Diagonal	Equal	Variable	Coordinate Axes
VVI	$\lambda_j A_j$	Diagonal	Variable	Variable	Coordinate Axes
EEE	λDAD^T	Ellipsoidal	Equal	Equal	Equal
EVE	λDA_jD^T	Ellipsoidal	Equal	Variable	Equal
VEE	$\lambda_j DAD^T$	Ellipsoidal	Variable	Equal	Equal
VVE	$\lambda_j DA_jD^T$	Ellipsoidal	Variable	Variable	Equal
EEV	$\lambda D_j AD_j^T$	Ellipsoidal	Equal	Equal	Variable
VEV	$\lambda_j D_j AD_j^T$	Ellipsoidal	Variable	Equal	Variable
EVV	$\lambda D_j A_j D_j^T$	Ellipsoidal	Equal	Variable	Variable
VVV	$\lambda_j D_j A_j D_j^T$	Ellipsoidal	Variable	Variable	Variable

Table 2.2: Types of Gaussian mixture models

"USArrests" dataset included in the R software, we can visualize the different types of models available and the resulting shape, volume, and orientation of the resulting Gaussian densities. The dataset itself provides the number of arrests per 100,000 residents in each of the 50 states in 1973 for three categories of crime: murder, assault, and rape. The dataset also contains the percent of the population living in urban areas. Each plot presented in Figure 2.2 to Figure 2.15 shows a Gaussian mixture model with 4 components where each colour represents a different cluster.

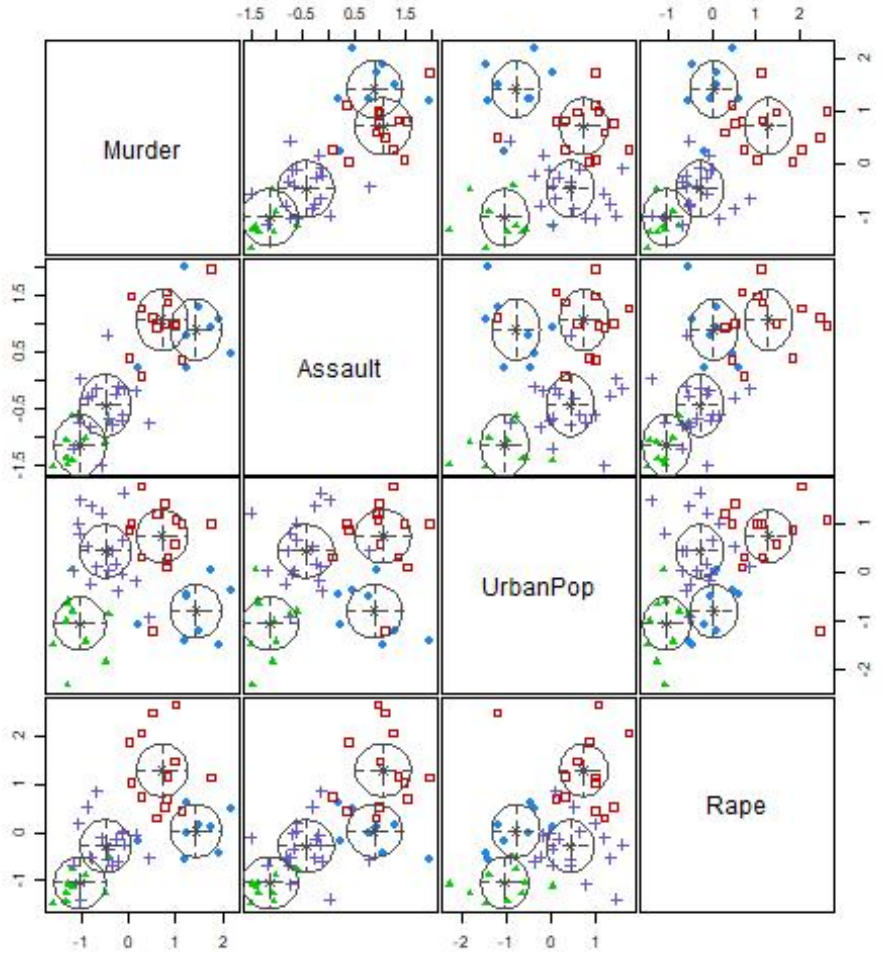

Figure 2.2: Gaussian mixture model: EII

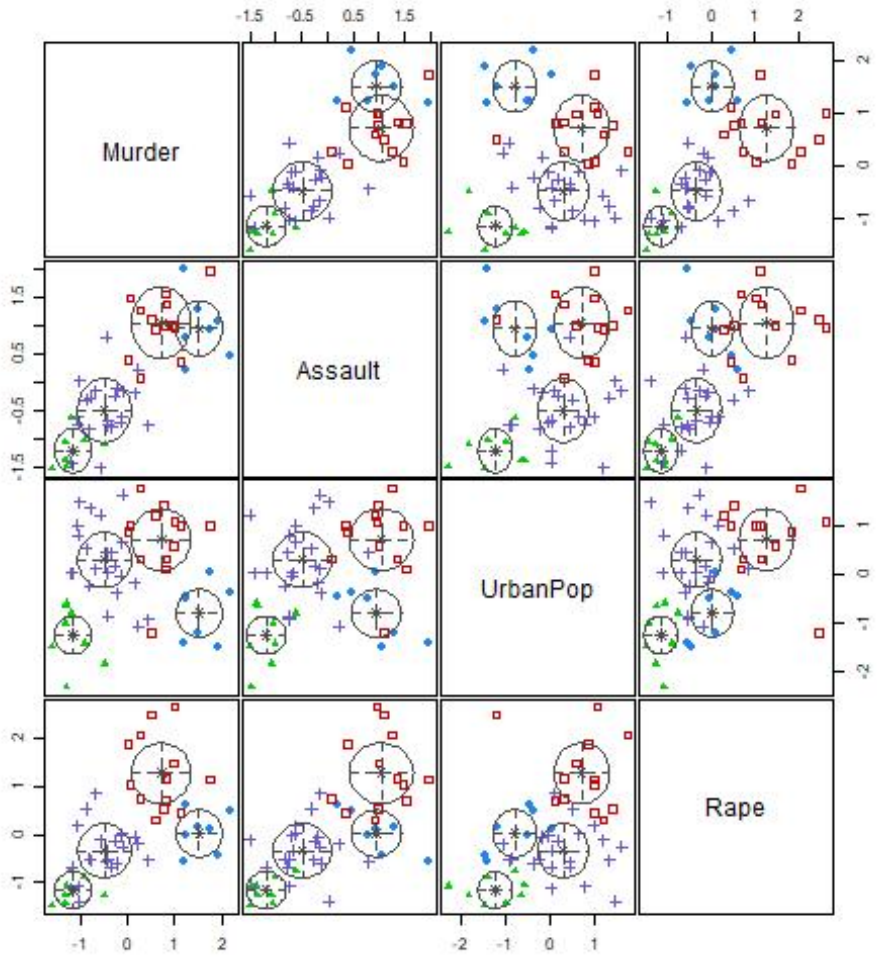

Figure 2.3: Gaussian mixture model: VII

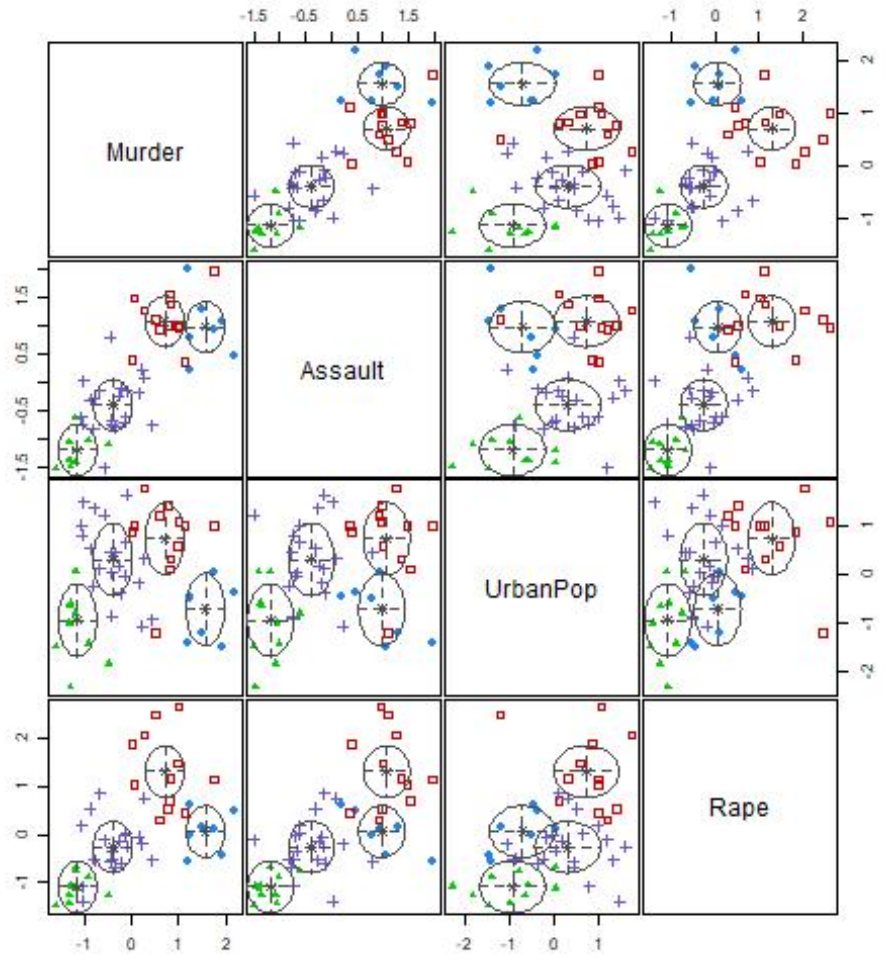

Figure 2.4: Gaussian mixture model: EEI

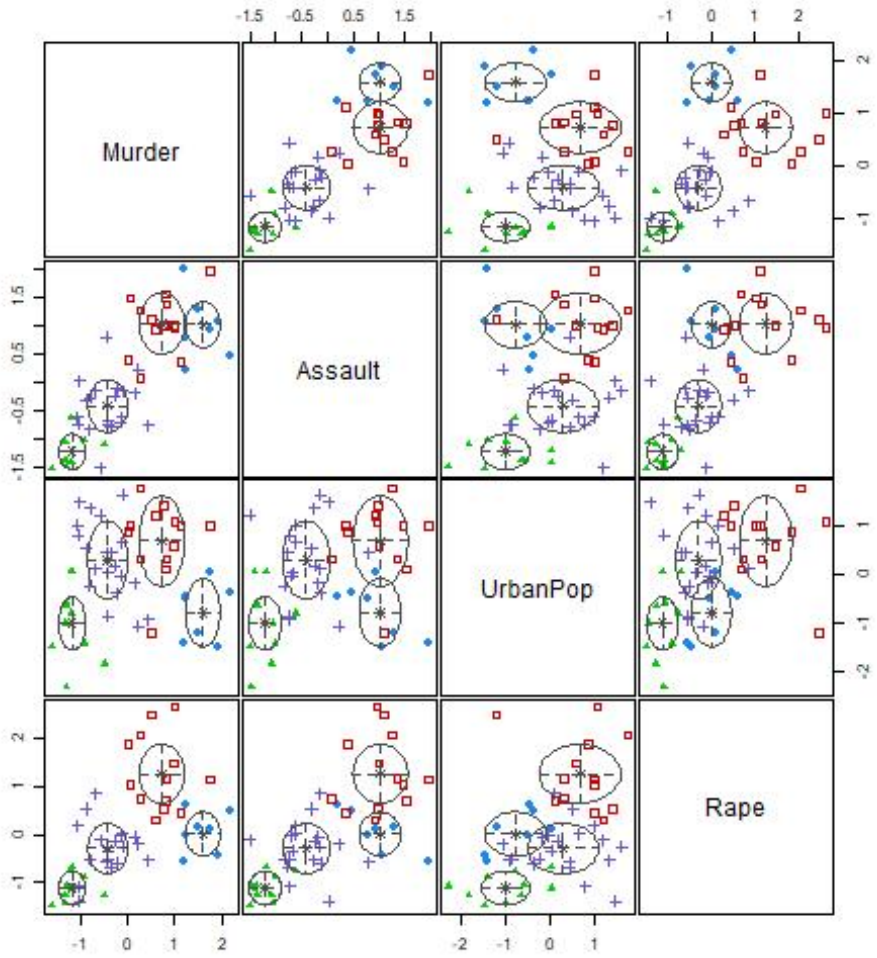

Figure 2.5: Gaussian mixture model: VEI

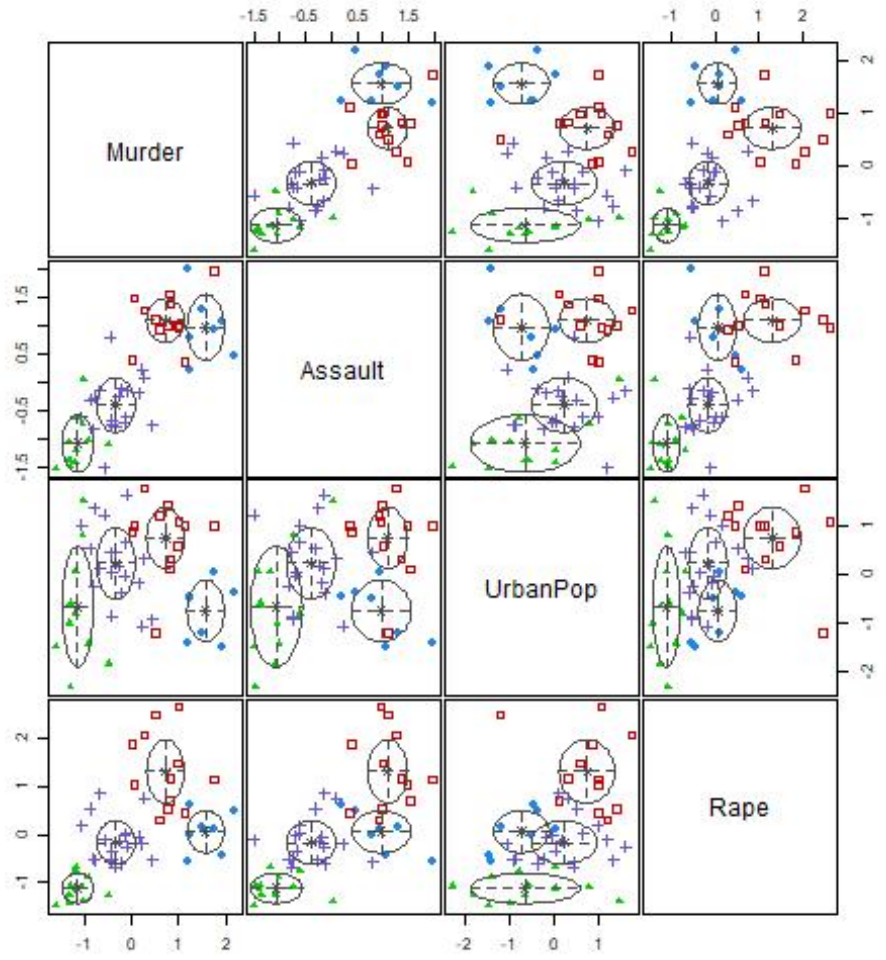

Figure 2.6: Gaussian mixture model: EVI

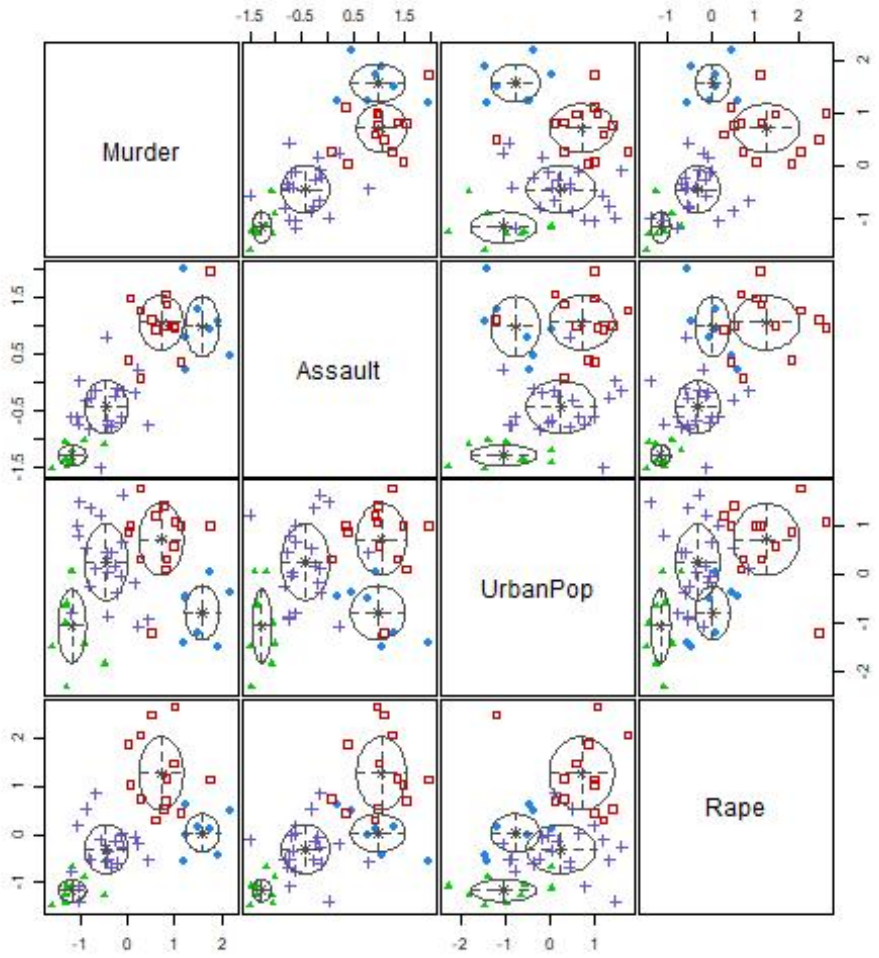

Figure 2.7: Gaussian mixture model: VVI

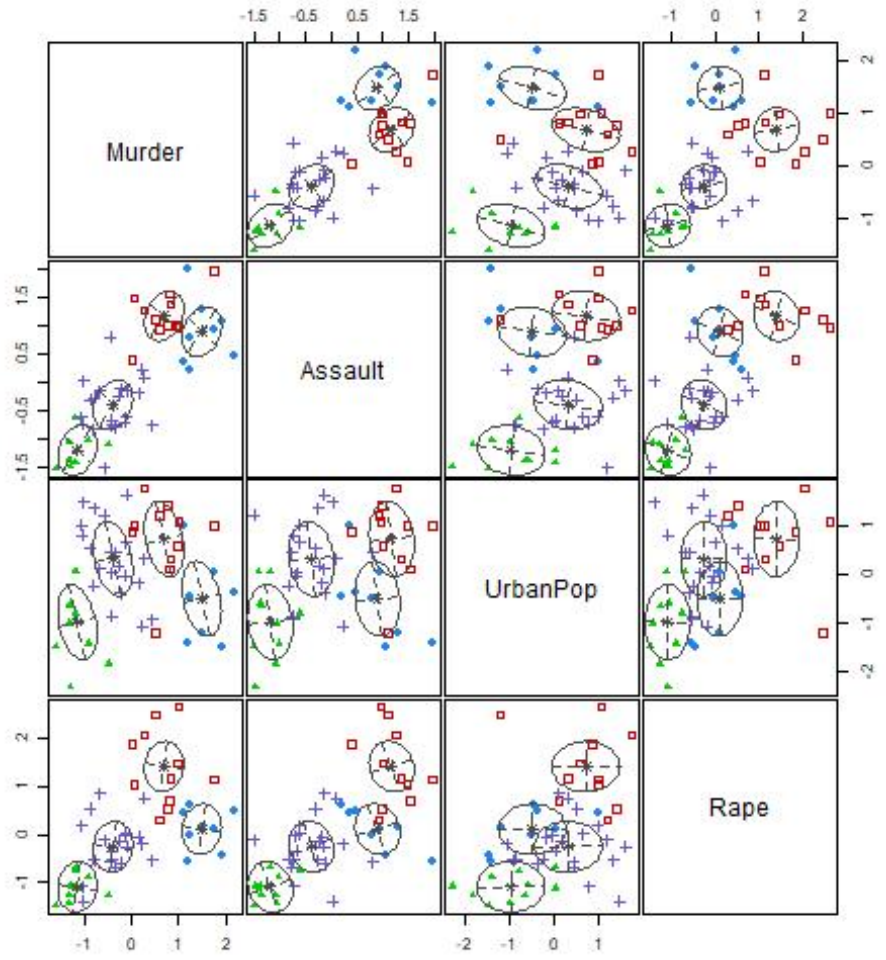

Figure 2.8: Gaussian mixture model: EEE

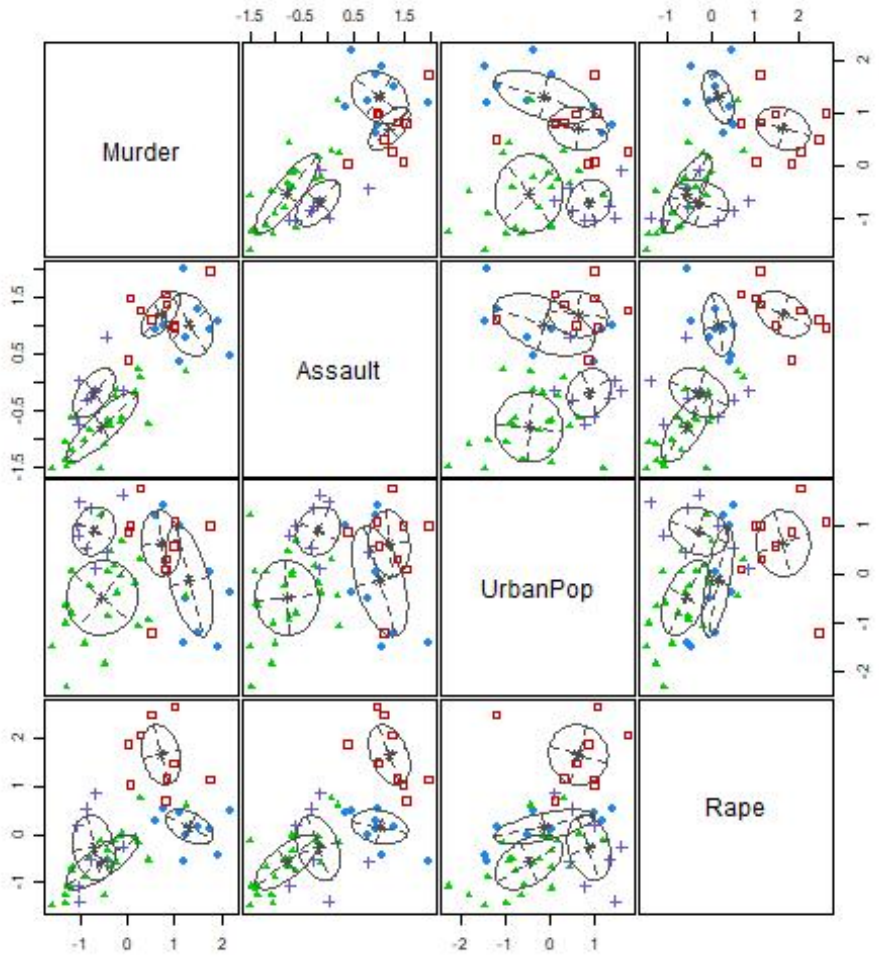

Figure 2.9: Gaussian mixture model: EVE

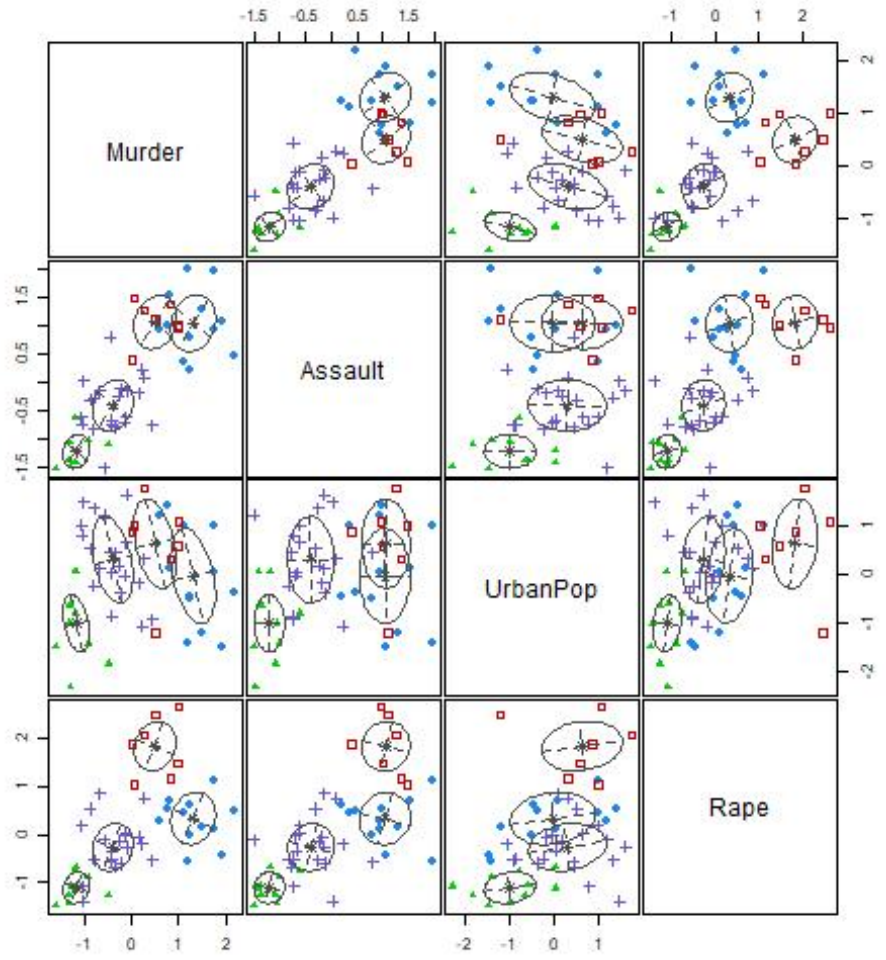

Figure 2.10: Gaussian mixture model: VEE

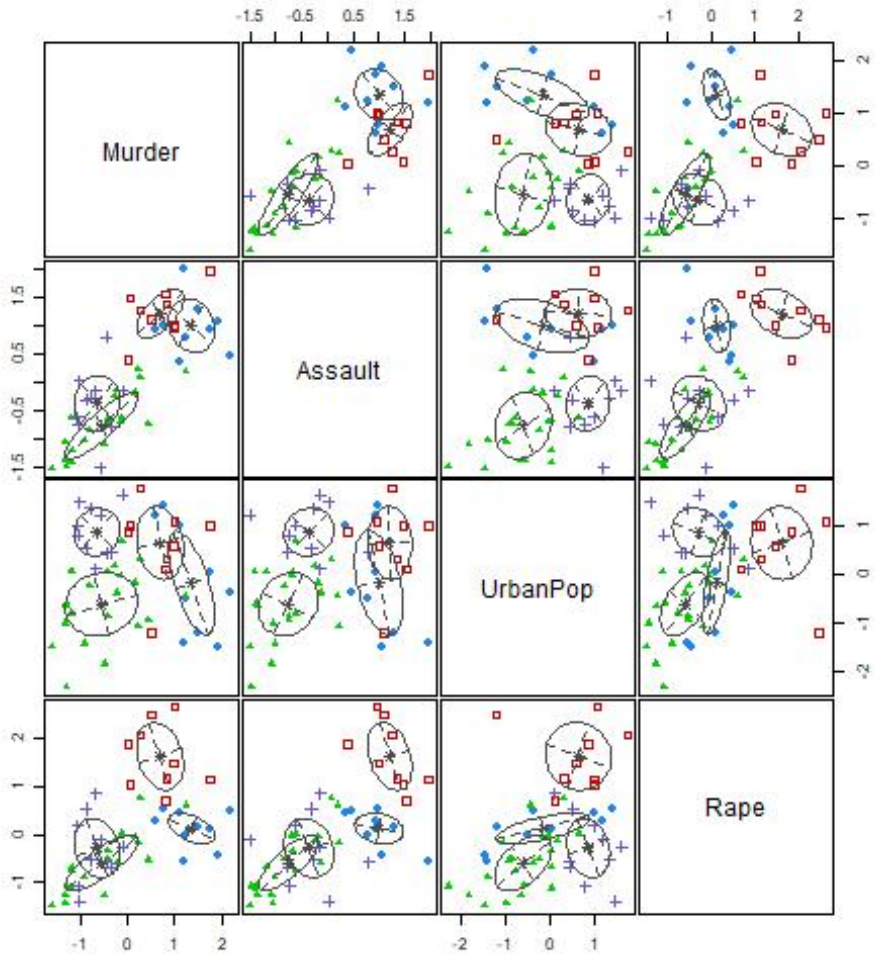

Figure 2.11: Gaussian mixture model: VVE

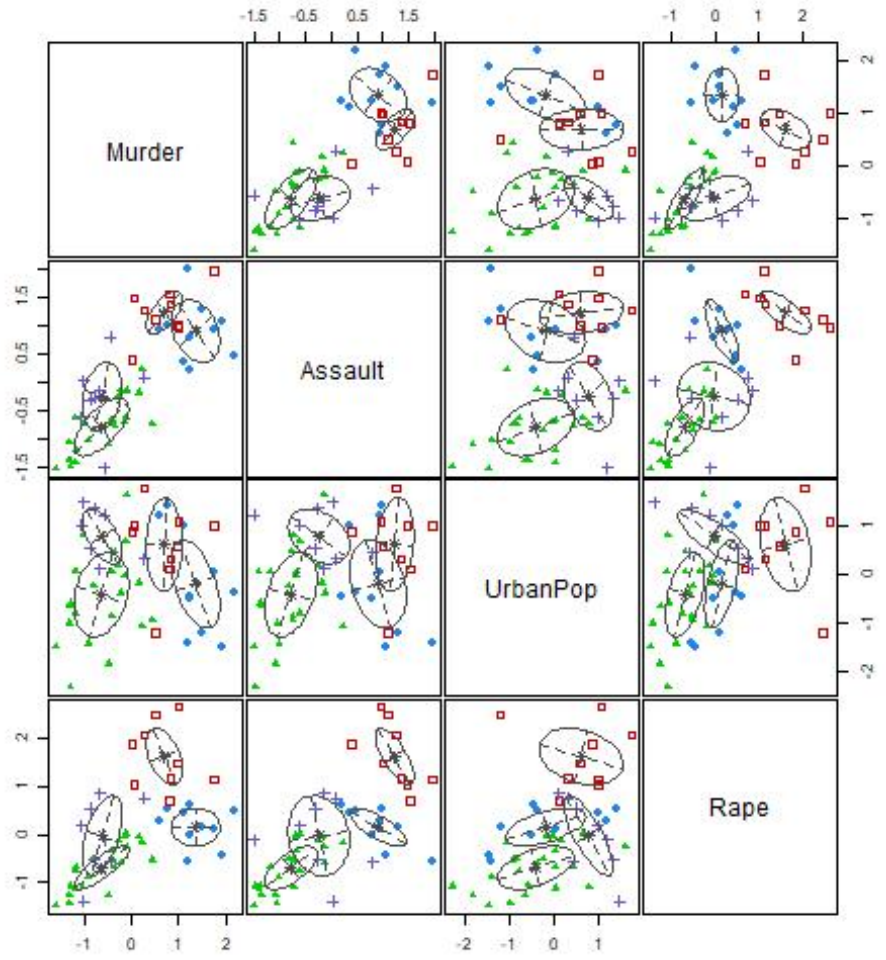

Figure 2.12: Gaussian mixture model: EEV

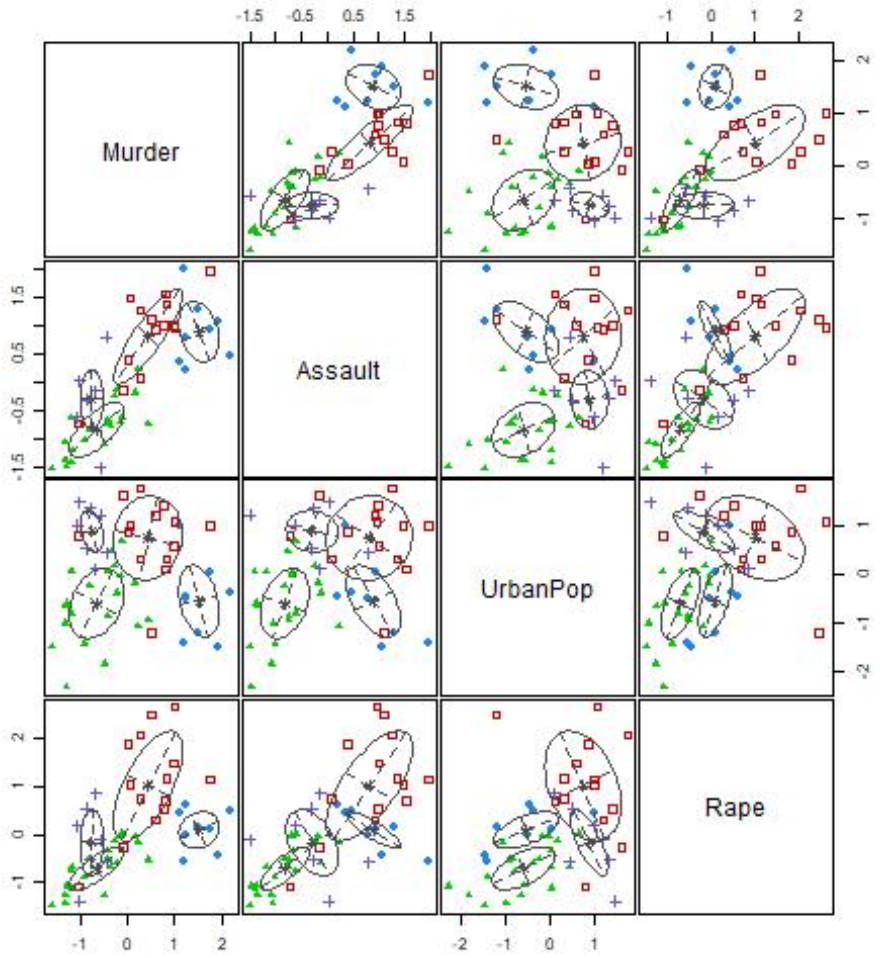

Figure 2.13: Gaussian mixture model: VEV

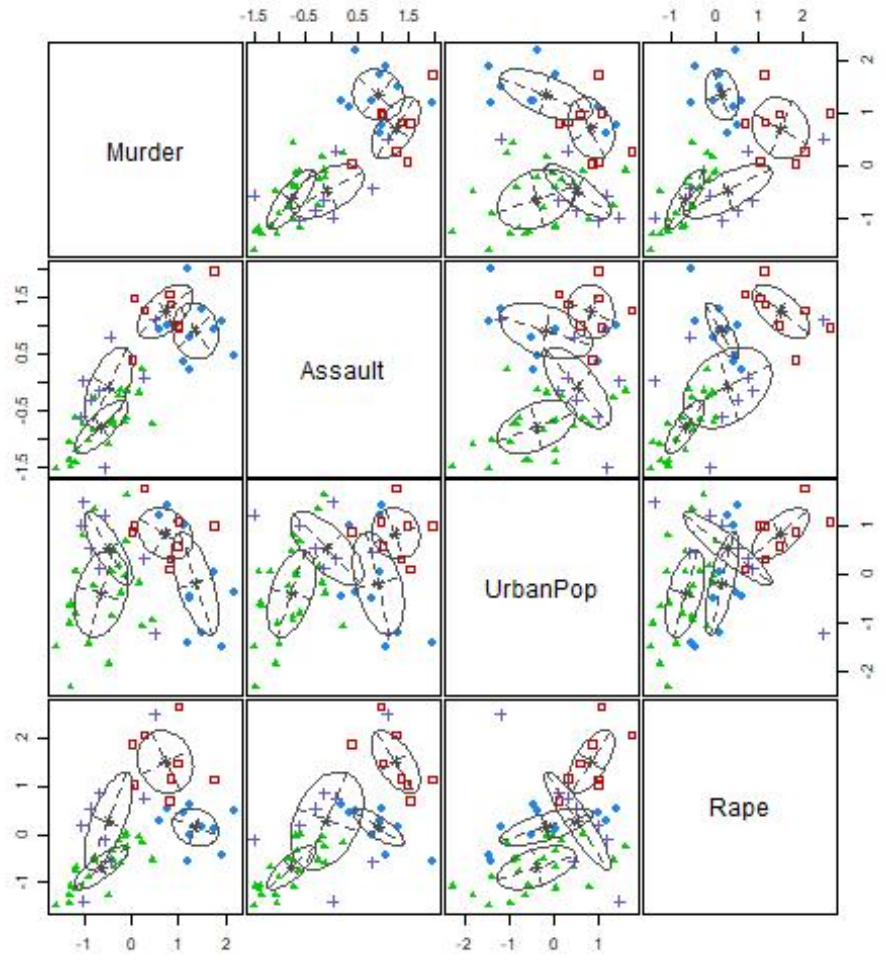

Figure 2.14: Gaussian mixture model: EVV

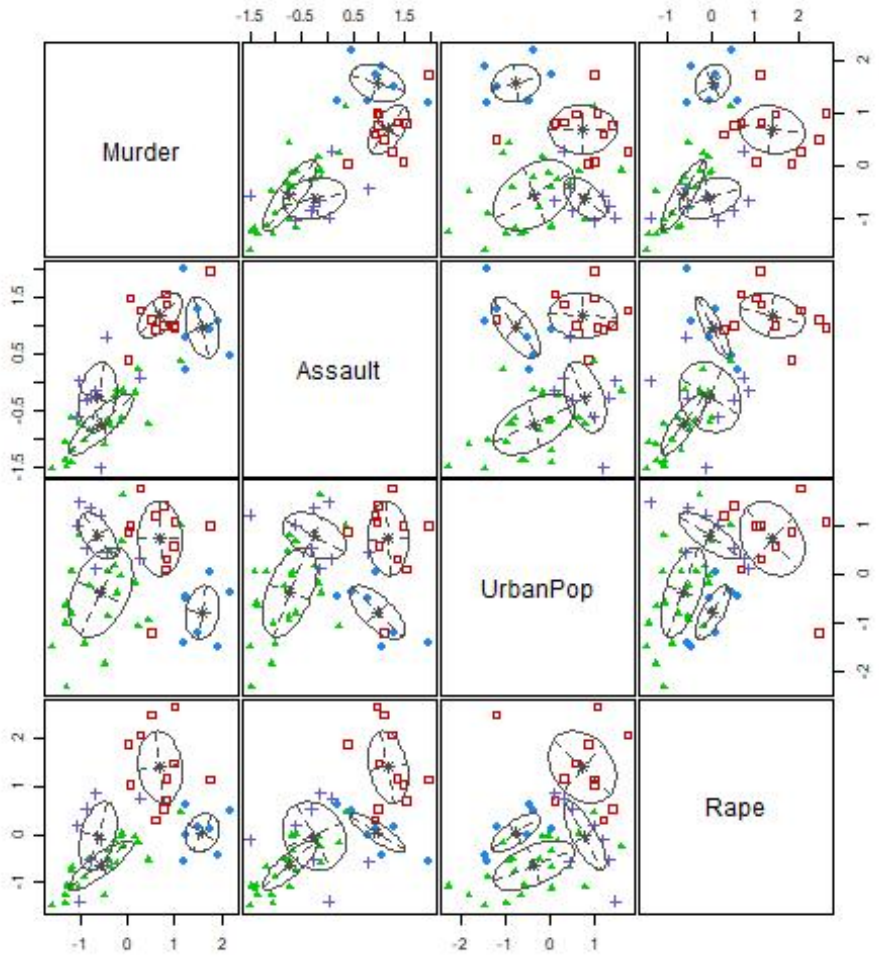

Figure 2.15: Gaussian mixture model: VVV

2.6.4 Bernoulli Mixture Model

Suppose that instead of continuous data, as in the Gaussian Mixture Model, we are concerned with binary data. That is, data comprised only of two categories such as the presence or absence of some quality. Similar to the GMM, we can fit a mixture of Bernoulli distributions to this data (Bishop, 2006). Recall that the general form of a mixture model is

$$f(x_i|\Theta) = \sum_{j=1}^{k} \pi_j f_j\left(x_i|\theta_j\right).$$

Now, our distribution, $f\left(x_i|\theta_j\right)$, will be a multivariate Bernoulli distribution with a D-dimensional probability vector, θ:

$$f\left(x|\theta\right) = \prod_{d=1}^{D} \left(\theta_d\right)^{x_d} \left(1-\theta_d\right)^{1-x_d}.$$

Then, the likelihood can be written as

$$L(\Theta|x_1,\ldots,x_n) = \prod_{i=1}^{n} f(x_i|\Theta),$$

where $\Theta = \{\pi_1,\ldots,\pi_k,\theta_1,\ldots,\theta_k\}$. We find the log-likelihood, ℓ, in the following manner:

$$\begin{aligned}\ell(\Theta|x_1,\ldots,x_n) &= \sum_{i=1}^{n} \log f(x_i|\Theta) \\ &= \sum_{i=1}^{n} \log\left[\sum_{j=1}^{k} \pi_j\, f\left(x_i|\theta_j\right)\right].\end{aligned}$$

Once again, this is problematic since this log-likelihood cannot be maximized by traditional means. Consequently, as in the Gaussian mixture model, we will introduce a latent variable denoted $z_i = \{z_{i1}, z_{i2}, \ldots, z_{ik}\}$. As a reminder, z_{ij} takes on the value 0 if x_i was not drawn from component j, and 1 if x_i was drawn from component j. Then, we can write the conditional distribution as

$$z_i \sim \text{Multinomial}(1, \boldsymbol{\pi}) \quad \text{where } \boldsymbol{\pi} = \{\pi_1, \ldots, \pi_k\}$$

$$f(x_i|z_i, \Theta) = \prod_{j=1}^{k} f_j(x_i|\theta_j)^{z_{ij}}.$$

As in the Gaussian mixture model, we can derive the complete log-likelihood:

$$\begin{aligned} f(x_i, z_i, |\Theta) &= f(x_i|z_i, \Theta) f(z_i|\Theta) \\ &= \prod_{j=1}^{k} f_j(x_i|\theta_j)^{z_{ij}} \prod_{j=1}^{k} \pi_j^{z_{ij}} \\ &= \prod_{j=1}^{k} (\pi_j f_j(x_i|\theta_j))^{z_{ij}} \\ L(\Theta|z_1, \ldots, z_n, x_1, \ldots, x_n) &= \prod_{i=1}^{n} f(x_i, z_i, |\Theta) \\ &= \prod_{i=1}^{n} \prod_{j=1}^{k} (\pi_j f_j(x_i|\theta_j))^{z_{ij}} \\ \ell(\Theta|z_1, \ldots, z_n, x_1, \ldots, x_n) &= \sum_{i=1}^{n} \sum_{j=1}^{k} z_{ij} \log \Big[\pi_j f_j(x_i, |\theta_j)\Big] \end{aligned}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{k} z_{ij} \log \left[\pi_j \prod_{d=1}^{D} (\theta_{jd})^{x_{id}} (1 - \theta_{jd})^{1-x_{id}} \right]$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{k} z_{ij} \left[\log \pi_j + \sum_{d=1}^{D} (x_{id} \log \theta_{jd} + (1 - x_{id}) \log (1 - x_{id})) \right].$$

The responsibility of a component given a data point can be found using Bayes' theorem, and is once again denoted τ_{ij}:

$$\tau_{ij} = \frac{f(x_i|\theta_d)\pi_j}{\sum_{l=1}^{k} f(x_i|\theta_l)\pi_l}.$$

Recall that τ_{ij} is computed in the E-step of the EM algorithm using the current values of the parameters. In the M-step, we update our parameters using τ_{ij}. In the Bernoulli mixture model, this is done as follows:

$$\pi_j^{\mathbf{new}} = \frac{1}{n} \sum_{i=1}^{n} \tau_{ij}$$

$$\theta_j^{\mathbf{new}} = \frac{\sum_{i=1}^{n} \tau_{ij} x_i}{\sum_{i=1}^{n} \tau_{ij}}.$$

Finally, we calculate the complete log-likelihood found earlier and check for convergence. If convergence has not been reached, we repeat the E and M steps. For further details, see (Bishop, 2006).

In R, the package we will use for fitting Bernoulli mixture model is called "flexmix" (Leisch, 2004), and uses the EM algorithm just discussed. There is another package, "BayesBinMix", (Papastamoulis and Rattray, 2017), that uses a Bayesian approach to fitting Bernoulli mixture models.

2.7 DBSCAN Clustering

Density-based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al., 1996) is a clustering algorithm that belongs to a larger family of algorithms known as density-based clustering. In density-based clustering, clusters are created in areas where observations are dense. There is no implicit or explicit assumptions about the shape or underlying distribution, unlike some of the techniques we have previously examined. As such, the shapes of the clusters can take on countless forms.

DBSCAN works with two parameters: ϵ and $MinPts$. The main challenge with practical usage of DBSCAN is tuning these parameters, as we will later see. The ϵ parameter defines the radius of an ϵ-neighbourhood around any given point p. This neighbourhood is denoted $N_\epsilon(p)$. If there are more than $MinPts$ points within $N_\epsilon(p)$, then p is considered a *core point*. That is, p is a core point if $|N_\epsilon(p)| \geq MinPts$. Furthermore, a point q is *directly-reachable* by a core point p if q is within $|N_\epsilon(p)|$. A point q is *reachable* by a core point p if there is a path of points, $p_1, p_2, \ldots, p_n$ such that p_{i+1} is directly-reachable by p_i for $1 \leq i \leq (n-1)$. Here, $p_1 = p$ and $p_n = q$. Then, since the definition of directly-reachable applies only to core points, it implies that all but potentially the last point, q, are core points (i.e. $p_1, \ldots, p_{n-1}$ are core points). If q is not a core point (i.e. $|N_\epsilon(q)| < MinPts$), but is within the ϵ-neighbourhood of some core point, then q is a *border point*. Finally, two points (not necessarily core points), p and q, are density-connected if there is a

common core point, s, such that p and q are both reachable from s.

With these definitions in mind, the algorithm for DBSCAN turns out to be relatively simple. Given a dataset $\mathcal{X} = \{x_1, \ldots, x_n\}$, the algorithm for DBSCAN is provided in algorithm 5.

Algorithm 5: DBSCAN clustering

1 For every point x_i in $\mathcal{X}$, find the points within $N_\epsilon(x_i)$. If $|N_\epsilon(x_i)| \geq MinPts$, record x_i as a core point.
2 For each core point, find all points that are reachable from that core point and assign them to the same cluster. Then, it is evident that any two points in the same cluster are density-connected.
3 If a point is neither a core point nor reachable from a core point, then assign it as noise (i.e. not belonging to any cluster).

The DBSCAN algorithm is advantageous in the sense that a pre-specified number of clusters is not required. Moreover, because the algorithm does not completely partition the dataset, DBSCAN is able to find observations that are considered noisy.

On the other hand, in datasets that have clusters of varied density, DBSCAN may struggle to find the appropriate clustering structure. Additionally, choosing an appropriate distance metric for ϵ requires careful consideration, especially in high-dimensions when the Euclidean distance may become an inappropriate choice.

2.7.1 Parameter-Tuning

As previously mentioned, the parameters ϵ and $MinPts$ are of great importance to the algorithm and require further consideration.

The $MinPts$ parameter is usually chosen based on domain knowledge. Generally, as the value of $MinPts$ increases, the number of clusters will decrease, as the requirement for a point to be considered a core point is stricter. Conversely, a low value of $MinPts$ will generally result in a higher number of clusters. With a lack of domain knowledge, a rule of thumb is to choose $MinPts$ to be equal to $2 * dim$ where dim is the dimension of the dataset (Sander et al., 1998).

The ϵ parameter is often chosen based on a plot known as the k-nearest neighbour plot (Schubert et al., 2017). The plot is constructed as follows:

1. For $k = MinPts$, calculate the distance to the nearest k neighbours for each data point in the dataset.
2. Take the maximum value of these distances (i.e. the distance from a point to its k-nearest neighbour) for each point.
3. Sort the values from step 2 from smallest to largest.
4. Plot the distances on the y-axis and an index on th x-axis.

Once this has been plotted, the distance on the y-axis that corresponds to an "elbow" in the plot will be taken to be an appropriate ϵ value. The elbow method works because if a value of ϵ is chosen that is too small, there will be an overwhelming amount of data points considered to be noise. On the other hand, if ϵ is chosen to be too large, it is possible that clusters will begin to merge together and too many data points are assigned to be in a given cluster. Therefore, by investigating the distance at which the elbow of the k-nearest neighbour distance plot occurs, we can hypothetically observe the ϵ value at

which this changeover occurs.

DBSCAN is implemented in R under the “dbscan” package (Hahsler et al., 2019).

Chapter 3

Clustering Evaluation

3.1 Clustering Tendency

Before applying clustering methods to a given dataset, the clustering *tendency* of the data should be measured. In other words, we seek to answer if there is any inherent clustering structures present in the data. One common way of assessing clustering tendency is known as the Hopkins statistic.

3.1.1 Hopkins Statistic

The Hopkins statistic is defined as follows (Hopkins and Skellam, 1954): suppose we have a dataset, $\mathbf{X}$, of dimension d and composed of n observations. To see if there are any clustering structures present in $\mathbf{X}$, it is a necessary condition that $\mathbf{X}$ is **not** uniformly distributed. Therefore, to test this condition, we will first take a random sample of m observations from $\mathbf{X}$, with $m < n$, denoted $\mathbf{S} = \{s_1, s_2, \ldots, s_m\}$. Then, we randomly generate m observations

with dimension d from a uniform distribution, denoted $\mathbf{Y} = \{y_1, y_2, \ldots, y_m\}$. For each $y_i \in \mathbf{Y}$, we find its nearest neighbour in the original dataset $\mathbf{X}$ and record the distance between the two points:

$$u_i = \min_{x \in \mathbf{X}} d(y_i, x),$$

where $d(\cdot)$ is a distance metric. Similarly, we record the distance between each s_i and its nearest neighbour in $\mathbf{X}$ (excluding the observation s_i):

$$w_i = \min_{x \in \mathbf{X}, x \neq s_i} d(s_i, x).$$

The Hopkins statistic is then defined as

$$H = \frac{\sum_{i=1}^{m} u_i^d}{\sum_{i=1}^{m} u_i^d + \sum_{i=1}^{m} w_i^d},$$

where the d in the exponent is the dimension of the dataset. The Hopkins statistic lies in the interval $(0, 1)$.

Consider a scenario in which $\mathbf{X}$ is uniformly distributed. Then, it is clear that $\sum_{i=1}^{m} u_i^d$ and $\sum_{i=1}^{m} w_i^d$ would be close to one another, and so H would be close to 0.5. Conversely, if $\mathbf{X}$ were **not** uniformly distributed, then $\sum_{i=1}^{m} u_i^d$ should be larger than $\sum_{i=1}^{m} w_i^d$. Therefore, in this scenario, H should be closer to 1.

The choice of m is crucial to the calculation of the Hopkins statistic. It has been suggested to choose m so that it is large enough to ensure the $2m$ nearest neighbour distances are statistically independent, but small enough

that certain distributional assumptions hold (namely, that the test statistic H follows a beta distribution with parameters m and m). A common value of m is chosen to be $m = 0.1n$.

In R, there are two functions that implement the Hopkins statistic. The first is the *hopkins* function from the "clustertend" (YiLan and RuTong, 2015) package. Note that when using this function, the Hopkins statistic is reported as $1 - H$ (where H is as defined in subsection 3.1.1). Thus, values closer to 0 indicate a highly clusterable dataset. The second function is the *get_clust_tendency* function from the "factoextra" (Kassambara and Mundt, 2020) package (the Hopkins statistic calculated from this function is the same as we defined earlier).

3.2 Internal Validation Metrics

Internal validation metrics is one of the two types of metrics used to assess the validity of clustering results. Internal metrics uses only information available within the dataset itself, as opposed to external validation metrics which uses external information such as class labels. Thus, internal metrics can be calculated on any given clustering output and are much more common in clustering analysis. Four of the most commonly-used internal validation metrics for clustering algorithms are explained in this section.

3.2.1 Dunn Index

Before introducing the Dunn Index, we must define a few variables used in the calculation of the index. Let m be the number of clusters, C_i and C_j be two particular clusters, δ be a measure of inter-cluster distance, and Δ be a measure of intra-cluster distance. δ and Δ can be defined, respectively, in many different ways. For instance, δ, the inter-cluster distance, can simply be defined as the distance between the centers of two particular clusters. Similarly, Δ, the intra-cluster distance can also be defined in a number of ways, such as the average distance between a point and the center of the cluster to which it belongs. Then, the Dunn Index, or DI, is defined as (Dunn, 1974)

$$DI_m = \frac{\min\limits_{1 \le i < j \le m} \delta(C_i, C_j)}{\max\limits_{1 \le k \le m} \Delta_k}.$$

That is, DI is the ratio of the smallest inter-cluster distance to the largest intra-cluster distance.

If we consider a clustering in which the individual clusters are compact and well-separated, then the numerator of DI will be large as a result of the separation between clusters. Conversely, the denominator will be small due to the compactness within the clusters. Therefore, DI has a value between 0 and infinity and ought to be maximized.

3.2.2 Silhouette Value

The silhouette value is defined as follows (Rousseeuw, 1987). Let C_i be a cluster, x be a point belonging to cluster C_i, and n_i be the number of total points in C_i. We define $a(x)$ as

$$a(x) = \frac{1}{n_i - 1} \sum_{y \in C_i, y \neq x} d(x, y),$$

where $d(x, y)$ is the (Euclidean, Manhattan, etc.) distance between x and y. Note that if $a(x)$ is relatively small, it means that point $x \in C_i$ is similar to other points in C_i. We can then define $b(x)$ as

$$b(x) = \min_{j, j \neq i} \left(\frac{1}{n_j} \sum_{y \in C_j} d(x, y) \right).$$

That is, $b(x)$ is the minimum mean distance between C_i and another cluster, C_j. C_j is determined by calculating the distance between point x and every point belonging to C_j, and then dividing by the number of points in C_j. Finally, the silhouette value for point x is defined as

$$s(x) = \begin{cases} \frac{b(x) - a(x)}{\max(a(x), b(x))}, & \text{if } n_i > 1 \\ 0, & \text{if } n_i = 1 \end{cases}.$$

If $a(x)$ is small, it means that the average distance between x and the points within the same cluster is small. On the other hand, if $b(x)$ is large, it means

that the average distance between x and the neighbouring cluster of C_j is large. As a result, x is a poor match to the neighbouring cluster, which is a desirable quality. Consequently, a small $a(x)$ and a large $b(x)$ will result in a silhouette value close to 1. Conversely, a small $b(x)$ and a large $a(x)$ will result in a silhouette value close to -1.

If the mean silhouette value is taken across all points within a cluster, it can be thought of as a measure of the compactness of the data in that cluster and how well the points belong to a given cluster. Values close to 1 indicate desirable clustering results, whereas values close to -1 indicate poor clustering.

3.2.3 Davies-Bouldin index

The Davies-Bouldin index is defined as follows (Davies and Bouldin, 1979). Let the scatter of cluster i be denoted S_i and defined by:

$$S_i = \frac{1}{n_i} \sum_{x \in C_i} d(x, c_i),$$

where c_i is the center of cluster i. This distance function can be, for example, the Euclidean distance between the point x and the centroid of the cluster. This scatter, then, is a measure of the compactness of C_i. Furthermore, let a measure of separation between two clusters be defined as

$$M_{i,j} = d(c_i, c_j).$$

Then, let $R_{i,j}$ be defined as

$$R_{i,j} = \frac{S_i + S_j}{M_{i,j}}.$$

In an ideal clustering scenario, the scatter of cluster i and cluster j is low (i.e. the clusters are compact) and the distance between these two clusters is large (i.e. the clusters are well-separated). That is, S_i and S_j are small and $M_{i,j}$ is large. Therefore, $R_{i,j}$ will be relatively small in a desirable clustering output.

Next, we define D_i as

$$D_i = \max_{j \neq i} R_{i,j}.$$

That is, we seek another cluster j which maximizes the quantity $R_{i,j}$. Note that this is the "worst-case" scenario. We repeat the process for each cluster in the partitioned data and take the average of D_i across all clusters. This is how we obtain the Davies-Bouldin index, defined as

$$DB = \frac{1}{m} \sum_{i=1}^{m} D_i.$$

Here, m is the total number of clusters obtained from the clustering algorithm. In summary, the Davies-Bouldin index considers the compactness within clusters and separation between clusters and should be minimized.

3.2.4 Connectivity Index

The connectivity index of a clustering partition, as its name implies, measures how "connected" a clustering output is. Here, connectedness is defined as the degree to which a clustering partition groups data points into the same cluster as their nearest neighbour(s). The connectivity index is defined as (Handl et al., 2005)

$$\text{Connectivity Index} = \sum_{i=1}^{N} \sum_{j=1}^{K} x_{i,nn_{i(j)}},$$

where N is the total number of data points in the dataset, K is a parameter that determines how many nearest neighbours to use in the calculation, $nn_{i(j)}$ is the j^{th} nearest neighbour of data point i, and $x_{i,nn_{i(j)}}$ is defined as

$$x_{i,nn_{i(j)}} = \begin{cases} \frac{1}{j}, & \text{if data points } i \text{ and } nn_{i(j)} \text{ belong to different clusters} \\ 0, & \text{if data points } i \text{ and } nn_{i(j)} \text{ belong to the same cluster} \end{cases}.$$

If a large number of the nearest neighbours of observations in the dataset belong to different clusters, then it is clear that the connectivity index will be large. Thus, the connectivity index takes on values between $[0, \infty]$ and should be minimized for valid clustering partitions.

In R, the connectivity index is implemented using the *connectivity* function in the "clValid" package (Brock et al., 2008). Note that the function uses a default of 10 nearest neighbours (i.e. $K = 10$).

3.3 External Validation Metrics

Unlike internal evaluation metrics, which are calculated solely based on the clustering partitions obtained from clustering algorithms, external evaluation metrics rely on external information. This external information is often in the form of class labels known beforehand (Palacio-Niño and Galiano, 2019). However, this is somewhat paradoxical, as clustering algorithms are often used in the absence of any ground truth. Therefore, external validation metrics are uncommon when validating clustering outputs. Nevertheless, if labels happen to be available, external validation metrics can be valuable in determining which candidate clustering partition is best suited for the data. Therefore, we will present the most common external validation metrics in this section.

3.3.1 Rand Index

Consider a dataset $\mathcal{X} = \{x_1, \ldots, x_n\}$ and two different clustering partitions of $\mathcal{X}$, denoted $U = \{u_1, \ldots, u_R\}$ and $V = \{v_1, \ldots, v_C\}$. Here, R and S are the respective number of clusters and need not be the same. Note that in practice, one of these clustering partitions is known to be correct (i.e. the ground truth) and will be used for assessing accuracy of the clustering algorithm. For example, the correct clustering partition can be assessed prior to analysis by a domain expert. The rand index (Rand, 1971) is a way of measuring the agreement between the clustering partitions of U and V. For example, consider any pair of elements of $\mathcal{X}$. There are four possible outcomes:

1. The pair is placed in the same cluster of U and the same cluster of V.
2. The pair is placed in two distinct clusters of U and two distinct clusters of V.
3. The pair is placed in the same cluster of U and two distinct clusters of V.
4. The pair is placed in two distinct clusters of U and the same cluster of V.

The first two outcomes above are considered to be agreements between U and V, whereas the last two would be disagreements. It is evident that agreements between U and V are desirable. The rand index is defined as the ratio of total agreements to the total number of agreements and disagreements:

$$R = \frac{\text{agreements}}{\text{agreements} + \text{disagreements}}.$$

Note that the denominator is simply the total number of outcomes (i.e. the total number of pairs). Therefore, this can be expressed as

$$R = \frac{\text{agreements}}{\binom{n}{2}}.$$

The rand index is defined on the interval $[0, 1]$ and larger values indicate a higher agreement between two clustering partitions.

3.3.2 Adjusted Rand Index

Consider the following motivating example for the Adjusted Rand Index (ARI): suppose that $\mathcal{X}$ is some dataset that contains n observations (where n is a large integer), U is a clustering partition on $\mathcal{X}$ composed of n partitions (each

partition contains 1 observation), and V is a different partition containing $n-1$ partitions (where all but one partition each contains one observation). If a pair of observations is randomly drawn from $\mathcal{X}$, then it is guaranteed that they belong to different cluster partitions of U, and it is extremely likely that they also belong to different partitions of V. Therefore, if the Rand index were calculated on U and V, it would be very close to 1 and so one may conclude that the clustering partitions agree very closely. Thus, just by increasing the number of clusters in U and V for a fixed n, the Rand index increases. The ARI attempts to solve this issue by correcting for chance.

The clustering partitions of U and V can be visualized as in the table below (Wikipedia contributors, 2021), where u_i is a particular cluster of U, v_j is a particular cluster of V, and n_{ij} is the number of observations that are shared between u_i and v_j.

	v_1	v_2	$\cdots$	v_C	sums
u_1	n_{11}	n_{12}	$\cdots$	n_{1s}	a_1
u_2	n_{21}	n_{22}	$\cdots$	n_{2C}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
u_R	n_{R1}	n_{R2}	$\cdots$	n_{RC}	a_R
sums	b_1	b_2	$\cdots$	b_C	

Then, the ARI is defined as (Hubert and Arabie, 1985)

$$ARI = \frac{\text{Index - Expected Index}}{\text{Max Index - Expected Index}}$$

$$= \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] \Big/ \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] \Big/ \binom{n}{2}}.$$

The ARI has a maximum value of 1 (indicating identical cluster partitions), and has a value of 0 when the Rand Index equals its expected value.

3.3.3 Mutual Information

Mutual Information (MI), like the Rand index, is a method of quantifying the similarity between two clustering partitions, U and V. There is also a corrected-for-chance version of the MI called the Adjusted Mutual Information (AMI), which will be described in the next section.

The RI and ARI belongs to a class of metrics that are calculated based on pair-counting. However, the MI metric belongs to a different class with its origin in information theory. Research has shown that the ARI may be an appropriate metric when clusters are large and approximately equal in size (Romano et al., 2016). Conversely, the AMI may be appropriate when clusters are small and potentially imbalanced.

Once again, let U and V be the two non-overlapping clustering partitions we wish to compare with R and C partitions, respectively. The *entropy* of a discrete random variable, X, that can assume values $\mathcal{X} = \{x_1, \ldots, x_n\}$, and with probability mass function $p(x)$, is defined as (Shannon, 1948)

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x).$$

The Mutual Information is a way of quantifying the amount of information shared between random variables X and Y. With $H(X,Y)$ denoting the joint entropy between X and Y, the MI is defined as (Shannon, 1948) (Kreer, 1957)

$$I(X,Y) = H(X) + H(Y) - H(X,Y)$$

$$= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}.$$

In the context of clustering, the MI is found as follows:
Let $P(i)$ denote the probability of an observation drawn randomly belonging to partition u_i of U. Then,

$$P(i) = \frac{|u_i|}{n},$$

where n is, again, the number of observations in our dataset and $|u_i|$ is the number of observations in partition u_i. Similarly,

$$P'(j) = \frac{|v_i|}{n}$$

would be the probability of an observation drawn randomly belonging to partition v_j of V. The entropy for U and V, according to the definitions above, would then be calculated as

$$H(U) = -\sum_{i=1}^{R} P(i) \log P(i)$$

$$H(V) = -\sum_{j=1}^{C} P'(j) \log P'(j).$$

The MI of U and V is then

$$I(X,Y) = \sum_{i=1}^{R}\sum_{j=1}^{C} P(i,j)\log\frac{P(i,j)}{P(i)P(j)},$$

where $P(i,j)$ is defined as the probability that an observation belongs to both cluster u_i and v_j:

$$P(i,j) = \frac{|u_i \cap v_j|}{n}.$$

The mutual information is a non-negative quantity, where a value of 0 indicates that no information is shared between two clustering partitions. On the other hand, the mutual information is bounded above by the entropy of U and V, and a higher value indicates more information in common between the two partitions (i.e. similar clusters).

3.3.4 Adjusted Mutual Information

The MI of two clustering partitions has the same issue that the Rand Index did. That is, as the number of clustering partitions in U and V increases for a fixed n, the MI will also increase. To adjust for chance, the AMI is corrected in a similar fashion:

$$AMI = \frac{\text{MI - Expected MI}}{\text{Max MI - Expected MI}}.$$

The actual formula for AMI is quite complicated, but is provided with a full explanation in (Xuan Vinh et al., 2010).

The AMI is bounded above by 1, indicating identical cluster partitions, and below by 0, which is when the MI between the two clustering partitions equals its expected value (Xuan Vinh et al., 2010).

3.4 Stability Metrics

Stability is an important concept in clustering solutions. The main idea is that once we have obtained a valid clustering partition, the structure of those clusters should not disappear when the input data is changed in a non-essential manner (Hennig, 2007). In other words, a new dataset that is similar to our original input dataset should produce similar clusters (Ben-David and Luxburg, 2008). However, there are a few questions that must be answered before we can define stability metrics. Namely,

1. How do we change the data in a non-essential way. That is, how do we obtain data that is different enough from the original data to assess stability, while ensuring that the underlying structure of the data is preserved?
2. How do we compare the two clustering partitions obtained from the original data and the "new" data?

These questions will be answered in the next two sections.

3.4.1 Resampling schemes

To assess stability, one must first obtain a new dataset that is similar to the original data. Because we are unaware of the true generating mechanism from which the data was obtained, or the structure of the true clusters, one must generate new data by perturbing the original data in some manner. Each of the resampling schemes described below does so in a different manner.

One of the most popular ways of obtaining new data is to generate non-parametric bootstrap samples from the original dataset Hennig (2007). The bootstrap samples are obtained by simple random sampling with replacement from the original dataset. That is, if the original dataset, $\mathcal{X}$ contains n points, points are repeatedly drawn from $\mathcal{X}$ until a new dataset is formed consisting of n points. This process is repeated for the appropriate number of bootstrap samples. Bootstrapping works because of the assumption that the sample taken, $\mathcal{X}$ is a good approximation of the true population. By generating new samples from the empirical distribution of $\mathcal{X}$, we can investigate the quality of inferences on this empirical distribution. Furthermore, our assumption that the original sample approximates the true population (which is reasonable for large sample sizes), allows us to draw an analogy between the inferences on the empirical distribution and the inferences on the true population. One of the disadvantages with bootstrapping methods is that by sampling with replacement, there may be repeated data points in the new dataset, which may result in the creation of new "mini-clusters". Alternatively, one may create

bootstrap samples and then simply remove any duplicate points.

To avoid multiple points, one may also create new subsets of the original data by sampling without replacement (Hennig, 2007). However, one must then choose an appropriate number of points, m, which is smaller than n. If m is too large, the resulting dataset may not be different enough from the original dataset. On the other hand, an m that introduces too much variation may result in clustering outputs that are entirely unrecognizable. Hennig suggests using the floor of $n/2$ for the value of m.

Another method of contaminating the data is to replace some observations by noise points Hennig (2007). These points may be allowed to lie far away from the existing data, or may be allowed to lie in between existing data to interfere with the separability of clusters (or both). This requires choosing the number of data points to replace and a distribution from which these noisy points should be drawn.

A somewhat similar method is to add noise to existing observations (Hennig, 2007). This is known as adding "jitter". Again, this involves choosing a distribution from which n errors, denoted e, are generated. The new data points, y_i, are then simply $y_i = x_i + e_i$ for $i = 1, \ldots, n$.

3.4.2 Comparing Original and Resampled Partitions

Once one has obtained a perturbed dataset by means of the previous of the section, we can compare clustering partitions from this dataset and the original

dataset.

One approach to comparing two clusters is to use the Jaccard similarity coefficient (Jaccard, 1901). Let A and B be two sets. The Jaccard similarity coefficient is defined as

$$\gamma(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

It is apparent that this $0 \leq \gamma(A, B) \leq 1$, with 0 implying that the sets share no points in common.

One approach to assessing cluster-wise stability is suggested by Hennig and requires the following steps.

Let S be the desired number of resampled datasets. Then, for $i = 1, \ldots, S$,

1. Obtain data using one of the resampling schemes from the previous section, where $\mathcal{X} = \{x_1, \ldots, x_n\}$ denotes the original dataset and $\mathcal{Y}_i = \{y_{i1}, \ldots, y_{im}\}$ is the new dataset. Note that n does not necessarily equal m.
2. Let $U = \{u_1, \ldots, u_k\}$ be the original clustering partition on $\mathcal{X}$ with k clusters. Induce a new clustering on $\mathcal{Y}_i$, denoted $V_i = \{v_{i1}, \ldots, v_{ik}\}$, also consisting of k clusters.
3. Let $\mathcal{Z}_i = \mathcal{X} \cap \mathcal{Y}_i$. Then, $\mathcal{Z}_i$ consists of the points that are both in the original dataset and the new dataset.
4. Let $C_i = U \cap \mathcal{Z}_i$. Furthermore, let $\Delta_i = V_i \cap \mathcal{Z}_i$. If C_i is nonempty, then compute the Jaccard similarity coefficient between C_i and each $D \in \Delta_i$

and record the largest Jaccard similarity coefficient. That is,

$$\gamma_{C,i} = \max_{D \in \Delta_i} \gamma\left(C_i, D\right).$$

On the other hand, if C_i is empty, then set $\gamma_{C,i} = 0$. In other words, for each cluster in the original partition, we are finding the cluster in the new partition that is most similar.

Once the above loop has completed and the maximum Jaccard coefficient has been recorded for each cluster in each iteration, we then calculate the mean similarity coefficient for each cluster across all iterations:

$$\bar{\gamma}_C = \frac{1}{S^*} \sum_{s=1}^{S} \gamma_{C,i},$$

for $1 \leq j \leq k$, where S^* is the number of new datasets for which C_i is nonempty. Therefore, if the mean similarity coefficient is close to 1 for a given cluster, then it is likely stable. On the other hand, values close to 0 indicate unstable clusters.

In summary, the above resampling methods and comparison methodology allows us to assess cluster-wise stability of a given clustering partition by measuring how likely we are to retrieve similar clusters given (slightly) perturbed data. The above algorithm is implemented in R using the *clusterboot* function in the "fpc" package (Hennig, 2020).

Chapter 4

Clustering Application to Casino Data

4.1 Description of Data

The original data provided by our local industry partner tracks individual user sessions on slot machines. This session data provides information such as a unique identification number for each patron, the date and time of the session, the duration of the session, and the number of coins entering (i.e. bets) the slot machine during a given player's session. These individual sessions were then aggregated into "casino visits", which allows us to examine the players' time spent in the casino and their gambling behaviour over the span of several months. In this context, a visit from a casino patron is comprised of one or more slot machine sessions from that particular patron. When an extended period of no slot machine activity is detected from a patron, the visit ends and a new visit begins the next time that player starts a slot machine session

at the casino. The clustering methods and evaluation metrics discussed in Chapter 2 and Chapter 3 are then applied to the visit data with the objective of identifying any discernible clusters of patrons based on their behaviour within the casino. The covariates shown in Table 4.1 were used in the cluster analysis.

Variable Name	Description
total_visits	The total number of times a player visited the casino in the time period of the dataset.
mean_duration	The mean length of time a player spent in the casino during their visits.
sd_duration	The standard deviation of time spent in the casino during a player's visits.
mean_games	The average number of games (spins) played on a slot machine during a player's visits.
sd_games	The standard deviation of games (spins) played on a slot machine during a player's visits.
mean_machines	The average number of different slot machines a player used during their visits.
sd_machines	The standard deviation of slot machines used during a player's visits.

Table 4.1: Description of covariates

4.2 Principal Components

Each covariate was scaled and principal components analysis was conducted on the dataset. The purpose of PCA was to simultaneously reduce the dimensionality of the data and construct pairwise uncorrelated covariates. In order to preserve the structure of the data, a cut off of 80% of the (cumulative) proportion of explained variance was used as a rule of thumb. In this manner,

the numbers of dimensions was reduced from the seven original covariates to three principal components that explain 85.7% of the variance as shown in Table 4.2.

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Standard deviation	1.868	1.235	0.994	0.768	0.533	0.306	0.159
Proportion of Variance	0.498	0.218	0.141	0.084	0.040	0.013	0.0036
Cumulative Proportion	0.498	0.716	0.857	0.942	0.983	0.996	1.000

Table 4.2: Proportion of variance explained by principal components

We can also examine a table of the PC loadings to discover the contributions of the covariates to each principal component. The loadings are shown in Table 4.3. The first three principal components will be used as covariates in

	PC1	PC2	PC3
total_visits	-0.07	0.12	0.98
mean_duration	-0.48	0.23	-0.09
sd_duration	-0.35	0.30	-0.17
mean_games	-0.48	0.14	0.00
sd_games	-0.47	0.18	0.02
mean_machines	-0.30	-0.63	0.03
sd_machines	-0.30	-0.62	0.05

Table 4.3: Principal Component loadings of each covariate

the subsequent sections for use in the clustering algorithms described above and are shown in Figure 4.1.

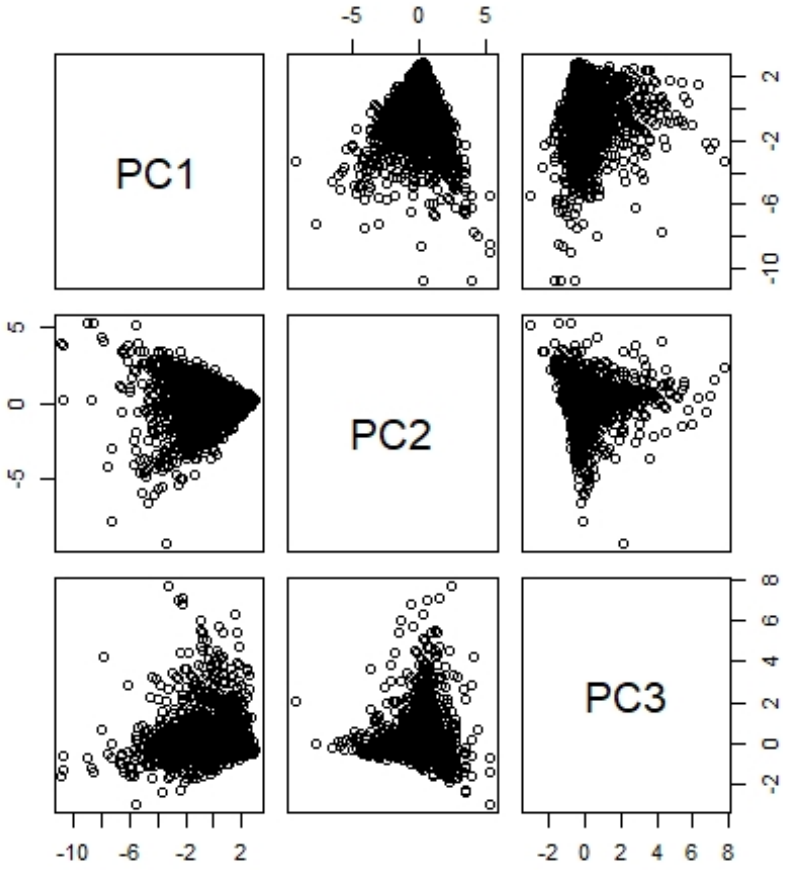

Figure 4.1: First three PC's of casino data

4.3 Measuring Cluster Tendency

To measure cluster tendency, we calculate the Hopkins statistic, detailed in subsection 3.1.1. The parameter necessary to calculate the statistic, m, is chosen to be $0.1n$. Since there are 2422 patrons in the dataset,we round up so

that $m = 25$.

Using the *get_clust_tendency* function, the Hopkins statistic is found to be

$$H = 0.9400715.$$

This indicates that the dataset is highly clusterable and we can therefore proceed with the clustering algorithms discussed in Chapter 2. It should be noted that a high value of the Hopkins statistic does not necessarily mean that clustering on a dataset will be *meaningful*. Instead, it can be interpreted as an indication that the dataset is not uniformly distributed and so one or more clusters is likely to be present within the data.

4.4 Application of Clustering Algorithms

One of the main challenges in many clustering algorithms is determining an appropriate number of clusters. We will determine this by varying the number of clusters in the algorithm and then computing the internal validation metrics discussed in Chapter 3: Dunn's index, silhouette value, Davies-Bouldin index, and connectivity index. In all evaluation methods, the Euclidean distance was chosen as our distance metric.

4.4.1 K-Means

Using the principal components calculated in the previous section as covariates, the results of the k-means clustering are shown in Table 4.4 for a pre-specified number of clusters between two and eight. That is, the number of clusters was

specified, the k-means clustering was performed, and evaluation metrics were calculated. The numbers in bold indicate the best value for each evaluation metric.

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	1.3066	0.0044	0.4145	**160.04**
3	1.2858	0.0037	0.4157	202.63
4	1.1441	0.0060	**0.4184**	242.77
5	1.0557	0.0056	0.3483	322.88
6	**1.0423**	**0.0071**	0.3478	371.53
7	1.0873	0.0049	0.3117	391.48
8	1.0806	0.0048	0.2959	464.26

Table 4.4: Results of k-means algorithm

Both the DB Index and Dunn's Index indicate that the ideal number of clusters is six, while the silhouette value and connectivity metric suggest a lower number of clusters. This discrepancy and problem of finding the "correct" number of clusters is a common issue that may arise when using a method such as k-means where the number of clusters needs to first be specified by the user without any ground truth.

4.4.2 K-Medoids

As in k-means clustering, k-medoids clustering requires that the user first specify a number of clusters. We proceed as before and calculate evaluation metrics to investigate the ideal number of clusters. Unlike k-means, however, k-medoids allows us to specify a metric other than the Euclidean distance. Therefore, we will use both the Euclidean and Manhattan (city-block) distance

to calculate the clustering output. Table 4.5 shows evaluation metrics for the k-medoids clustering using Euclidean distance, and Table 4.6 shows evaluation metrics for the k-medoids clustering using Manhattan distance.

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	1.3171	**0.0057**	**0.3849**	**177.23**
3	1.4671	0.0032	0.3470	303.82
4	1.4225	0.0034	0.2655	372.19
5	1.1472	0.0040	0.2907	358.57
6	1.1238	0.0046	0.2723	435.45
7	**1.0865**	0.0042	0.2889	509.06
8	1.1259	0.0021	0.2668	530.55

Table 4.5: Results of k-medoids algorithm using Euclidean distance

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	1.3197	0.0029	**0.3744**	**202.43**
3	1.6074	0.0029	0.2597	330.36
4	1.4341	**0.0036**	0.2637	348.09
5	1.1580	0.0036	0.2903	402.48
6	**1.1163**	0.0034	0.2691	438.70
7	1.1445	0.0033	0.2704	511.65
8	1.1333	0.0026	0.2965	542.08

Table 4.6: Results of k-medoids algorithm using Manhattan distance

In both cases, the DB index indicates that a higher number of clusters should be preferred, while the other metrics indicate a lower number of clusters is ideal.

4.4.3 Agglomerative Hierarchical Clustering

Unlike k-means and k-medoids, hierarchical clustering does not require the user to pre-specify a number of clusters before the output is provided. Instead, we must choose a particular distance metric and linkage method that will be used by the algorithm. For the distance metric, we use both the Euclidean and Manhattan metrics. We try three different linkage methods for each distance metric - single-linkage, complete-linkage, and average-linkage, all of which are described in Table 2.1. For each combination, a different number of clusters is chosen and once again the evaluation metrics are computed.

Table 4.7 through Table 4.12 provides the computed evaluation metrics for each combination of linkage method and distance metric.

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	0.2449	**0.2484**	**0.7036**	**2.93**
3	**0.2389**	0.1908	0.6975	5.86
4	0.2667	0.1693	0.6165	8.90
5	0.2619	0.1542	0.6180	14.75
6	0.3208	0.1609	0.6177	17.77
7	0.3092	0.1590	0.6145	21.20
8	0.3704	0.1767	0.6168	24.55

Table 4.7: Results of agglomerative hierarchical clustering using single-linkage and Euclidean distance

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	1.1958	0.0125	0.4159	**98.89**
3	0.9925	0.0147	**0.4268**	109.91
4	**0.9158**	0.0156	0.4096	120.72
5	1.0212	0.0186	0.3341	157.83
6	0.9705	0.0190	0.3080	166.27
7	1.0931	**0.0210**	0.2791	199.85
8	1.0964	0.0111	0.3148	303.98

Table 4.8: Results of agglomerative hierarchical clustering using complete-linkage and Euclidean distance

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	**0.4767**	**0.1908**	**0.7108**	**5.73**
3	0.5922	0.0713	0.6288	22.26
4	0.8348	0.0461	0.5077	63.07
5	0.8711	0.0539	0.4675	87.19
6	0.8633	0.0539	0.4645	88.42
7	0.7225	0.0539	0.4635	88.54
8	0.7545	0.0539	0.4610	92.84

Table 4.9: Results of agglomerative hierarchical clustering using average-linkage and Euclidean distance

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	0.2449	**0.2484**	**0.7036**	**2.93**
3	**0.2389**	0.1908	0.6975	5.86
4	0.2667	0.1693	0.6165	8.90
5	0.3080	0.1488	0.6203	15.27
6	0.3843	0.1488	0.6198	18.60
7	0.3635	0.1488	0.6169	21.53
8	0.2913	0.1309	0.6161	23.86

Table 4.10: Results of agglomerative hierarchical clustering using single-linkage and Manhattan distance

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	0.8255	0.0318	**0.5345**	**39.39**
3	**0.7832**	0.0332	0.5004	64.82
4	1.1006	**0.0426**	0.4860	88.29
5	1.3259	0.0092	0.2871	250.39
6	1.2106	0.0102	0.2820	257.32
7	1.1552	0.0108	0.2797	263.05
8	1.0986	0.0108	0.2746	263.05

Table 4.11: Results of agglomerative hierarchical clustering using complete-linkage and Manhattan distance

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	**0.2449**	**0.2484**	**0.7036**	**2.93**
3	0.5399	0.0365	0.5892	23.71
4	0.6028	0.0339	0.5149	50.69
5	0.7330	0.0407	0.4742	84.60
6	0.8131	0.0459	0.4550	87.92
7	0.8089	0.0473	0.4446	94.84
8	0.8053	0.0473	0.4255	100.22

Table 4.12: Results of agglomerative hierarchical clustering using average-linkage and Manhattan distance

4.4.4 Divisive Hierarchical Clustering

In divisive hierarchical clustering we can decide to use the Euclidean or Manhattan distance metric to compute dissimilarity, but there are no linkage methods involved. Thus, Table 4.13 shows metrics computed by performing divisive hierarchical clustering using the Euclidean distance, while Table 4.14 uses the Manhattan distance.

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	1.2265	0.0082	**0.4792**	**123.08**
3	1.0988	0.0103	0.4580	150.92
4	1.0189	0.0127	0.4221	167.89
5	**0.8505**	0.0133	0.4135	167.89
6	0.9175	0.0139	0.3793	183.01
7	1.0305	0.0163	0.3443	263.30
8	1.0014	**0.0166**	0.3440	269.04

Table 4.13: Results of divisive hierarchical clustering using Euclidean distance

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	1.2402	0.0063	**0.4825**	**134.63**
3	1.0997	0.0079	0.4636	169.98
4	1.0247	0.0089	0.4286	184.91
5	1.0946	0.0098	0.3921	202.02
6	1.2145	0.0098	0.3313	288.16
7	1.0813	**0.0120**	0.3289	288.16
8	**0.9788**	0.0120	0.3285	291.09

Table 4.14: Results of divisive hierarchical clustering using Manhattan distance

4.5 Fuzzy c-means

For the fuzzy c-means algorithm, a distance metric and a pre-specified number of clusters must be provided. As before, we used both the Euclidean and Manhattan distance, while varying the number of clusters. For the purposes of our analysis, we have varied the value the fuzzy parameter m.

Recall that fuzzy c-means results in a soft clustering output. That is, each patron will have some degree of membership in every cluster. Thus, the evaluation metrics we have used so far are calculated by assuming that a patron belongs to the cluster in which they have the largest degree of membership. This is an assumption that may not hold true in reality and eliminates some of the usefulness of the fuzzy c-means algorithm. In other words, we are "de-fuzzying" the soft clustering output and transforming it into a hard clustering output.

Table 4.15 to Table 4.27 shows evaluation metrics for the fuzzy c-means algorithm using Euclidean distance and a variety of values for the fuzzy parameter m. When the fuzzy parameter was above 2.2, the algorithm was unable to distinguish to which cluster each patron should belong. In other words, the clusters became too fuzzy.

Table 4.28 to Table 4.37 shows evaluation metrics for the fuzzy c-means algorithm using Manhattan distance and a variety of values for m. For values of m above 1.9, the clustering output once again became too fuzzy.

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	1.3066	0.0044	0.4145	**160.04**
3	1.2858	0.0037	**0.4157**	202.63
4	1.2967	0.0052	0.3381	287.67
5	1.3223	**0.0090**	0.3457	362.75
6	**1.0423**	0.0071	0.3478	371.10
7	1.0792	0.0076	0.3140	422.54
8	1.0933	0.0076	0.3083	464.85

Table 4.15: Results of fuzzy c-means clustering using Euclidean distance and $m = 1.0$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	1.3082	0.0025	0.4104	**186.57**
3	1.3050	0.0052	**0.4134**	212.55
4	1.2982	0.0048	0.3247	315.97
5	1.0576	0.0056	0.3475	326.90
6	**1.0428**	**0.0071**	0.3481	379.42
7	1.0878	0.0049	0.3115	403.64
8	1.0811	0.0015	0.2958	485.21

Table 4.16: Results of fuzzy c-means clustering using Euclidean distance and $m = 1.1$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	1.3096	**0.0059**	**0.4068**	**180.64**
3	1.3566	0.0052	0.3977	241.88
4	1.3259	0.0034	0.3103	342.53
5	1.0650	0.0031	0.3398	361.77
6	**1.0556**	0.0041	0.3436	404.14
7	1.1043	0.0049	0.3065	431.07
8	1.1365	0.0049	0.3110	458.78

Table 4.17: Results of fuzzy c-means clustering using Euclidean distance and $m = 1.2$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	1.3118	0.0029	**0.4005**	**174.01**
3	1.4329	0.0068	0.3693	281.49
4	1.3761	**0.0066**	0.2940	377.75
5	1.1332	0.0030	0.3182	399.84
6	**1.0703**	0.0044	0.3182	417.60
7	1.1158	0.0049	0.2939	530.82
8	1.1100	0.0051	0.3042	474.64

Table 4.18: Results of fuzzy c-means clustering using Euclidean distance and $m = 1.3$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	1.3127	0.0034	**0.3960**	**172.86**
3	1.5246	0.0028	0.3408	292.25
4	1.4219	0.0042	0.2803	400.16
5	1.5801	0.0051	0.2568	480.76
6	1.5696	0.0052	0.2560	545.72
7	**1.1503**	0.0032	0.2657	541.33
8	1.2161	**0.0057**	0.2417	541.55

Table 4.19: Results of fuzzy c-means clustering using Euclidean distance and $m = 1.4$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	**1.3134**	0.0029	**0.3907**	**181.07**
3	1.6405	0.0039	0.3059	309.98
4	1.4847	0.0044	0.2659	407.65
5	1.8328	**0.0051**	0.2248	504.48
6	1.8588	0.0036	0.2112	642.55
7	1.5042	0.0019	0.1988	592.51
8	1.5742	0.0028	0.1810	696.72

Table 4.20: Results of fuzzy c-means clustering using Euclidean distance and $m = 1.5$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	**1.3143**	**0.0049**	**0.3853**	**173.36**
3	1.7553	0.0038	0.2740	337.27
4	1.5814	0.0037	0.2472	380.24
5	2.2512	0.0020	0.1981	532.28
6	2.8106	0.0046	0.1798	671.99
7	2.9532	0.0026	0.1583	798.18
8	4.1523	0.0019	0.1419	862.89

Table 4.21: Results of fuzzy c-means clustering using Euclidean distance and $m = 1.6$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	**1.3147**	**0.0058**	**0.3817**	**190.27**
3	1.8487	0.0050	0.2440	318.35
4	2.0445	0.0038	0.2021	464.44
5	2.6726	0.0041	0.1710	581.75
6	3.4242	0.0029	0.1500	734.21
7	3.8039	0.0020	0.1343	841.21
8	4.2745	0.0019	0.1268	892.75

Table 4.22: Results of fuzzy c-means clustering using Euclidean distance and $m = 1.7$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	**1.3148**	0.0016	**0.3798**	**186.44**
3	1.9314	0.0022	0.2195	318.99
4	2.8886	**0.0041**	0.1840	475.10
5	3.0265	0.0014	0.1528	601.27
6	3.4004	0.0032	0.0898	667.29
7	3.8277	0.0022	0.1199	772.39
8	4.0814	0.0016	0.0774	837.21

Table 4.23: Results of fuzzy c-means clustering using Euclidean distance and $m = 1.8$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	**1.3153**	**0.0057**	**0.3748**	**191.93**
3	2.0143	0.0017	0.1994	357.02
4	3.6885	0.0017	0.1503	460.70
5	3.1546	0.0017	0.1514	493.44
6	3.0742	0.0016	0.1458	537.26
7	4.2856	0.0026	0.1105	640.17
8	3.7303	0.0041	0.1033	660.61

Table 4.24: Results of fuzzy c-means clustering using Euclidean distance and $m = 1.9$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	**1.3149**	**0.0054**	**0.3719**	**192.04**
3	2.0829	0.0030	0.1821	384.40
4	2.0514	0.0021	0.1811	367.64
5	2.0286	0.0029	0.1826	335.45
6	3.8914	0.0037	0.1394	508.46
7	3.5366	0.0035	0.1371	543.31
8	3.2921	0.0032	0.1327	509.99

Table 4.25: Results of fuzzy c-means clustering using Euclidean distance and $m = 2.0$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	**1.3149**	0.0029	**0.3703**	**193.58**
3	2.1460	0.0029	0.1650	358.77
4	2.1329	0.0020	0.1656	371.63
5	2.0944	0.0049	0.1639	355.43
6	2.0745	0.0041	0.1696	338.81
7	2.0714	**0.0050**	0.1695	334.76
8	4.1642	0.0037	0.1260	530.95

Table 4.26: Results of fuzzy c-means clustering using Euclidean distance and $m = 2.1$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	**1.3137**	0.0041	**0.3678**	**196.21**
3	2.2542	0.0043	0.1498	354.81
4	2.2203	**0.0051**	0.1478	362.44
5	2.1903	0.0016	0.1479	373.03
6	2.1473	0.0016	0.1464	387.10
7	2.1409	0.0016	0.1486	389.15
8	2.1451	0.0050	0.1462	361.83

Table 4.27: Results of fuzzy c-means clustering using Euclidean distance and $m = 2.2$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	1.3065	0.0044	**0.4148**	**162.20**
3	1.3637	0.0051	0.3901	240.55
4	1.3400	0.0049	0.3124	348.85
5	1.3167	0.0063	0.3378	384.08
6	1.1493	0.0043	0.3026	464.75
7	**1.1053**	0.0036	0.3028	478.10
8	1.1379	**0.0069**	0.3065	493.29

Table 4.28: Results of fuzzy c-means clustering using Manhattan distance and $m = 1.0$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	1.3081	0.0047	**0.4110**	**164.57**
3	1.6221	0.0041	0.3033	298.22
4	1.3310	0.0049	0.3112	378.90
5	1.3413	0.0052	0.3253	401.02
6	**1.0480**	0.0053	0.3553	439.22
7	1.1026	0.0040	0.3021	475.58
8	1.1496	**0.0074**	0.3051	492.39

Table 4.29: Results of fuzzy c-means clustering using Manhattan distance and $m = 1.1$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	1.3107	0.0041	**0.4041**	**182.82**
3	1.3282	**0.0065**	0.3994	249.19
4	1.3482	0.0025	0.3007	397.14
5	1.3720	0.0030	0.3160	451.17
6	**1.0604**	0.0053	0.3458	428.84
7	1.1222	0.0019	0.2967	499.15
8	1.1233	0.0041	0.3014	544.38

Table 4.30: Results of fuzzy c-means clustering using Manhattan distance and $m = 1.2$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	1.3120	0.0043	**0.3991**	**179.68**
3	1.4183	**0.0070**	0.3744	272.53
4	1.4049	0.0044	0.2825	422.66
5	1.1808	0.0032	0.3111	422.19
6	**1.0690**	0.0062	0.3299	438.12
7	1.1618	0.0034	0.2751	572.02
8	1.1478	0.0019	0.2775	596.72

Table 4.31: Results of fuzzy c-means clustering using Manhattan distance and $m = 1.3$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	1.3133	**0.0053**	**0.3930**	**176.09**
3	1.7324	0.0050	0.2758	335.86
4	1.4876	0.0025	0.2670	400.07
5	1.6700	0.0044	0.2411	597.61
6	**1.2666**	0.0033	0.2421	579.79
7	1.3254	0.0034	0.2317	583.42
8	1.3345	0.0046	0.2209	599.36

Table 4.32: Results of fuzzy c-means clustering using Manhattan distance and $m = 1.4$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	**1.3141**	**0.0057**	**0.3866**	**177.00**
3	1.8151	0.0021	0.2509	334.29
4	1.5697	0.0025	0.2539	379.16
5	2.3156	0.0044	0.2008	574.10
6	2.8671	0.0036	0.1755	741.05
7	2.5300	0.0030	0.1588	814.66
8	2.0662	0.0026	0.1551	832.44

Table 4.33: Results of fuzzy c-means clustering using Manhattan distance and $m = 1.5$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	**1.3147**	**0.0058**	**0.3819**	**192.63**
3	1.8823	0.0033	0.2239	307.41
4	3.0634	0.0041	0.1773	561.27
5	2.8361	0.0038	0.1596	701.45
6	2.7536	0.0025	0.1100	729.15
7	3.5486	0.0030	0.1606	859.70
8	4.1193	0.0021	0.0737	1001.45

Table 4.34: Results of fuzzy c-means clustering using Manhattan distance and $m = 1.6$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	**1.3153**	**0.0016**	**0.3782**	**203.56**
3	1.9637	0.0021	0.1954	342.37
4	3.6629	0.0029	0.1520	486.98
5	4.3956	0.0050	0.1408	606.98
6	4.4001	0.0023	0.0318	671.45
7	4.8229	0.0021	0.1104	769.33
8	5.9282	0.0048	0.1021	784.14

Table 4.35: Results of fuzzy c-means clustering using Manhattan distance and $m = 1.7$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	**1.3159**	0.0029	**0.3721**	**190.68**
3	2.0770	0.0041	0.1651	370.35
4	3.8464	0.0029	0.1186	365.18
5	4.3921	0.0023	0.1326	521.88
6	3.8806	**0.0049**	0.1227	529.60
7	3.9024	0.0034	0.0982	558.28
8	5.3970	0.0023	0.1001	633.26

Table 4.36: Results of fuzzy c-means clustering using Manhattan distance and $m = 1.8$

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
2	**1.3160**	**0.0041**	**0.3695**	**191.39**
3	2.1485	0.0041	0.1335	344.41
4	2.1140	0.0022	0.1327	366.64
5	2.1095	0.0024	0.1358	360.17
6	2.0699	0.0021	0.1318	344.12
7	2.0815	0.0016	0.1344	350.50
8	3.3861	0.0016	0.0855	400.27

Table 4.37: Results of fuzzy c-means clustering using Manhattan distance and $m = 1.9$

4.6 DBSCAN

To implement the DBSCAN clustering algorithm, the two parameters of k and ϵ must be chosen. As discussed in section 2.7, a common choice of k is chosen to be twice the dimension of the dataset. Here, we are using three principal components as our covariates, and so we will $k = 6$. We will then use this k for the k-nearest neighbour plot, shown in Figure 4.2.

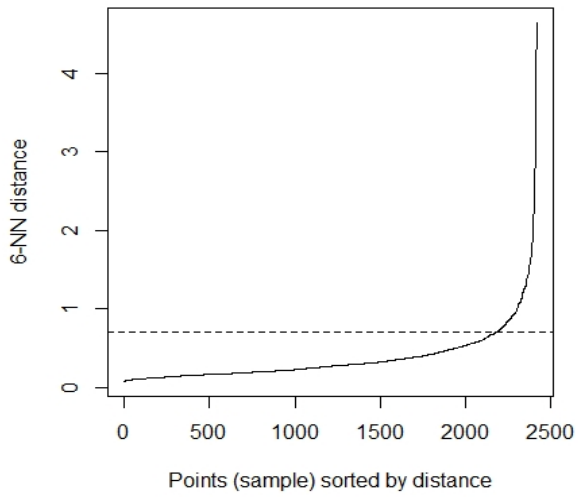

Figure 4.2: K-nearest neighbour plot ($k = 6$)

According to the plot, a reasonable value of ϵ is 0.7, which is indicated by the elbow of the plot. The dashed horizontal line within the plot shows this value.

With $k = 6$ and $\epsilon = 0.7$, the DBSCAN algorithm found three distinct clusters. The evaluation metrics for the output are shown in Table 4.38.

Clusters	DB Index	Dunn's Index	Silhouette Value	Connectivity
3	0.5920	0.0769	0.3996	136.13

Table 4.38: Results of DBSCAN clustering with $k = 6$ and $\epsilon = 0.7$

4.7 Gaussian Mixture Model

When fitting a Gaussian mixture model to the data, there is some randomness involved in terms of estimating parameters due to initialization and updates of the EM algorithm. As a result, to identify an optimal number of clusters, one run of the EM algorithm is insufficient and may be misleading. Therefore, we will initialize and run the algorithm several times and plot the output of the optimal number of clusters based on the BIC. Figure 4.3 shows the plot, where the horizontal axis represents iterations of the EM-algorithm and the vertical axis shows the optimal number of clusters. As can be seen, between 11 and 13 clusters seems to be ideal.

Next, we must decide if any constraints should be placed upon the covariance matrix. By plotting the BIC of all different available models (see Table 2.2), we can determine which fits best to the data. Figure 4.4 shows the BIC for all available models on the vertical axis, while the horizontal axis shows the number of components for the GMM (between 11 and 13 based on our previous analysis).

The best model, then would either be not constraining the covariance matrix at all (VVV), or constraining the covariance matrix to to have equal orientation (VVE), while allowing the volume and shape to vary.

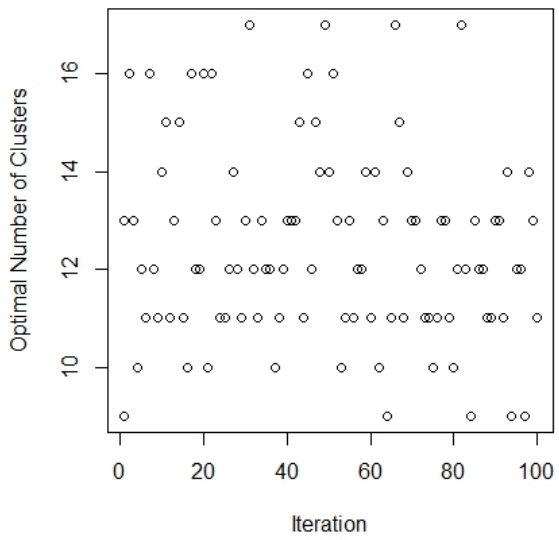

Figure 4.3: Optimal Number of Clusters for a Gaussian Mixture Model

Therefore based on these results, we will limit our analysis to Gaussian mixture models between 11 and 13 clusters with either no constraint on the covariance, or constraining the orientation of the covariance.

The evaluation metrics are shown in Table 4.39 for the six different combinations of cluster numbers and model types. To calculate these metrics, as in fuzzy c-means, we assign patrons to the cluster with the highest probability of membership. Again, this will eliminate some of the flexibility of soft clustering outputs by transforming it into a hard clustering and is only done for the purpose of analysis.

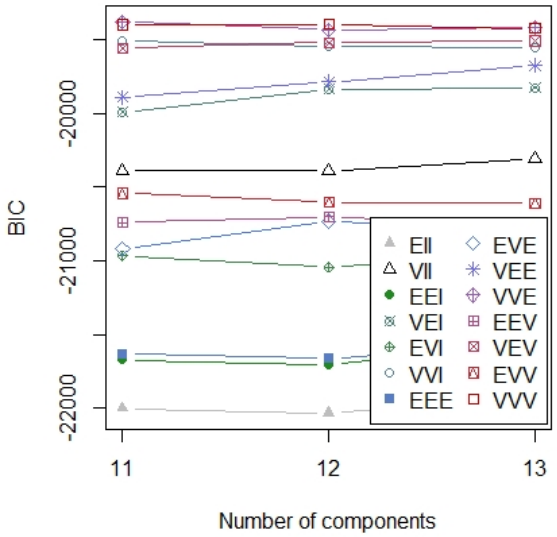

Figure 4.4: BICs of all Gaussian mixture models

Clusters	Model	DB Index	Dunn's Index	Silhouette Value	Connectivity
11	VVV	**1.608**	**0.002**	0.058	1138.60
11	VVE	1.854	0.001	0.065	**1084.34**
12	VVV	1.904	0.000	-0.001	1517.80
12	VVE	2.297	0.002	0.040	1298.75
13	VVV	2.008	0.001	0.025	1548.01
13	VVE	1.773	0.002	**0.072**	1267.09

Table 4.39: Evaluation Metrics for Gaussian Mixture Models

Additionally, the classification can be visualized using a pairs plot with the first three principal components, shown in Figure 4.5 through Figure 4.10. The plots also show the shape and orientation of the Gaussian densities. Notice the difference in orientation for the two different types of models presented.

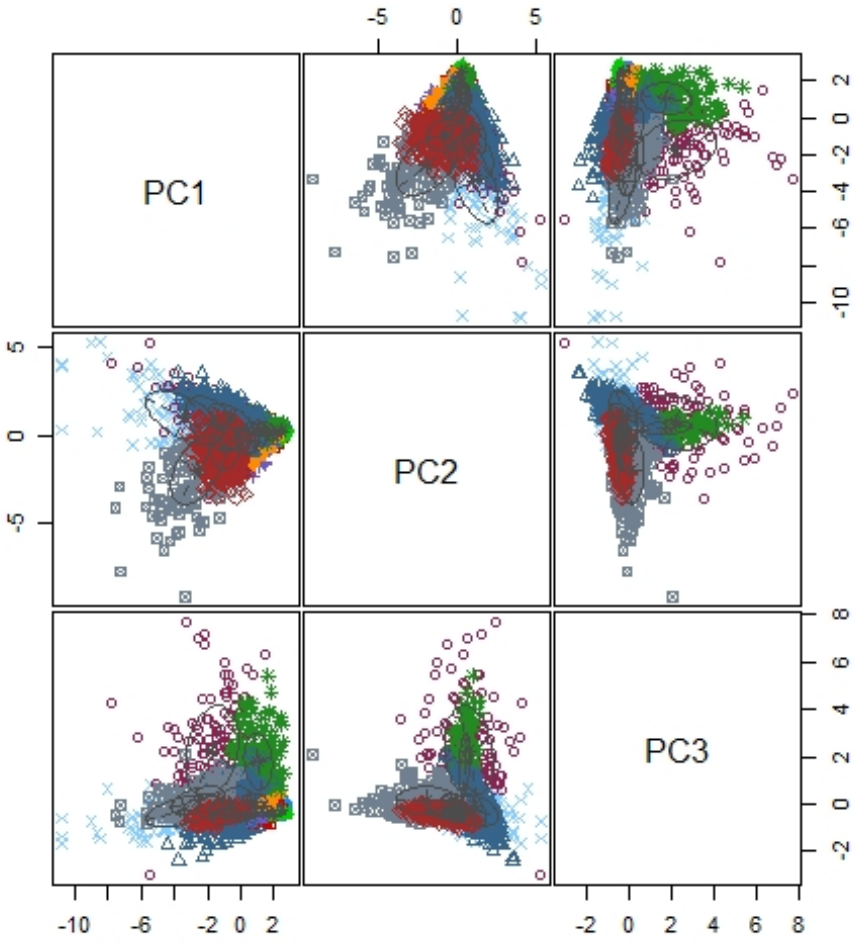

Figure 4.5: Gaussian Mixture Model - 11 components, Model type: VVV

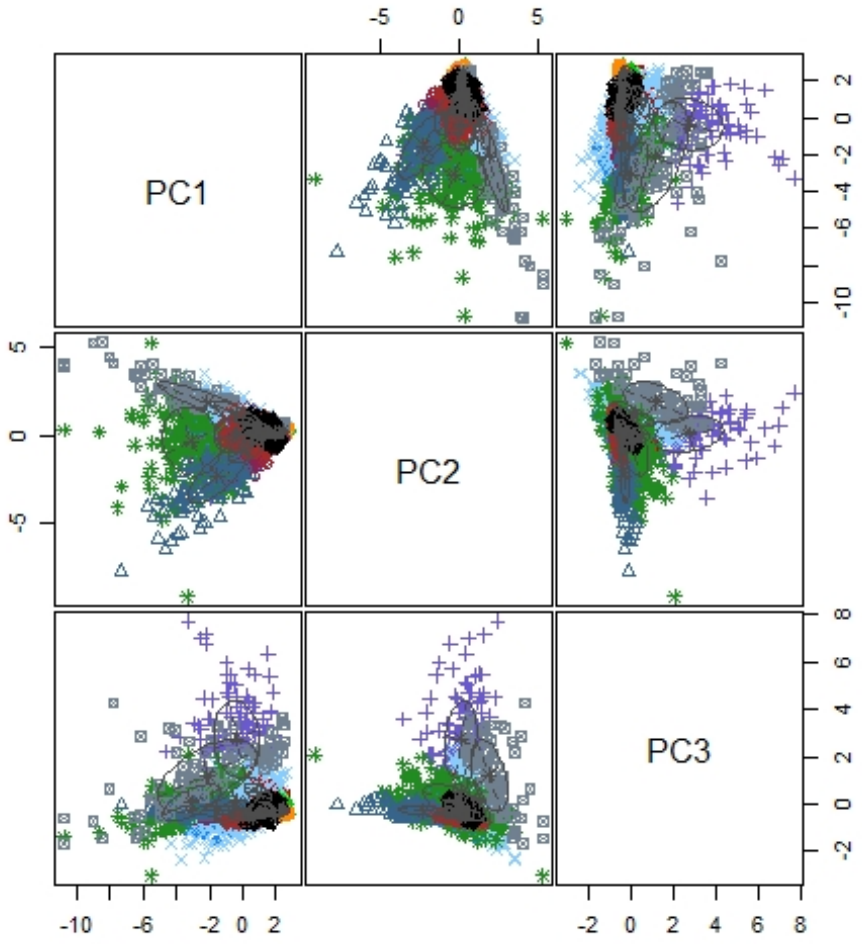

Figure 4.6: Gaussian Mixture Model - 12 components, Model type: VVV

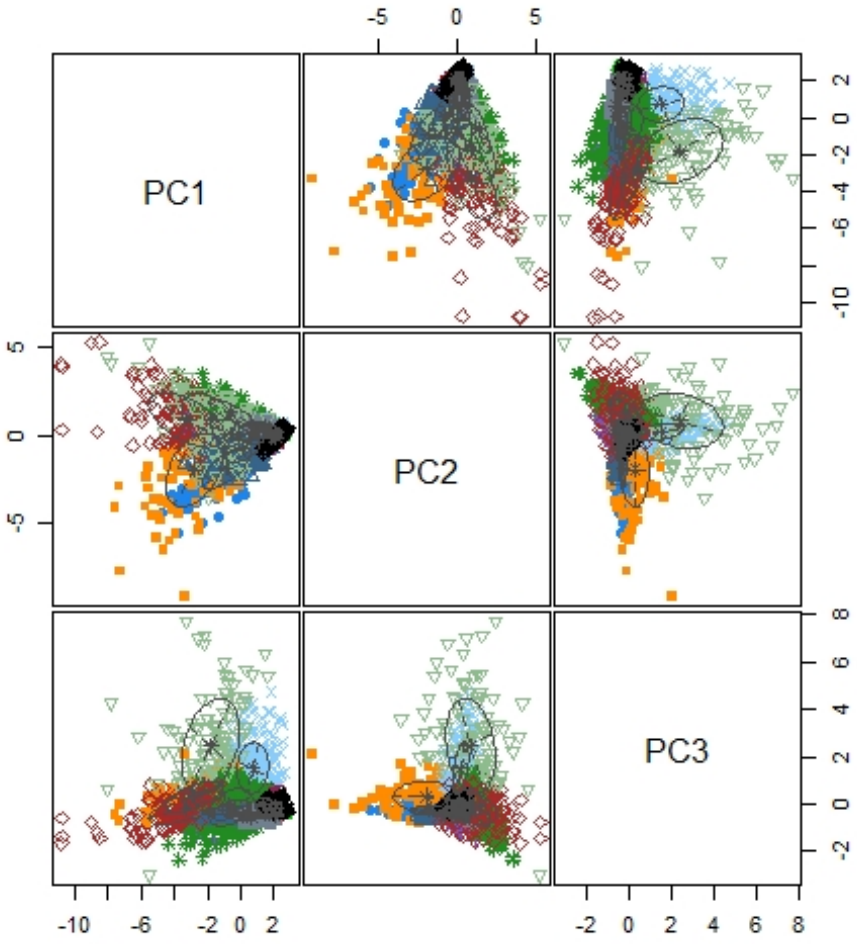

Figure 4.7: Gaussian Mixture Model - 13 components, Model type: VVV

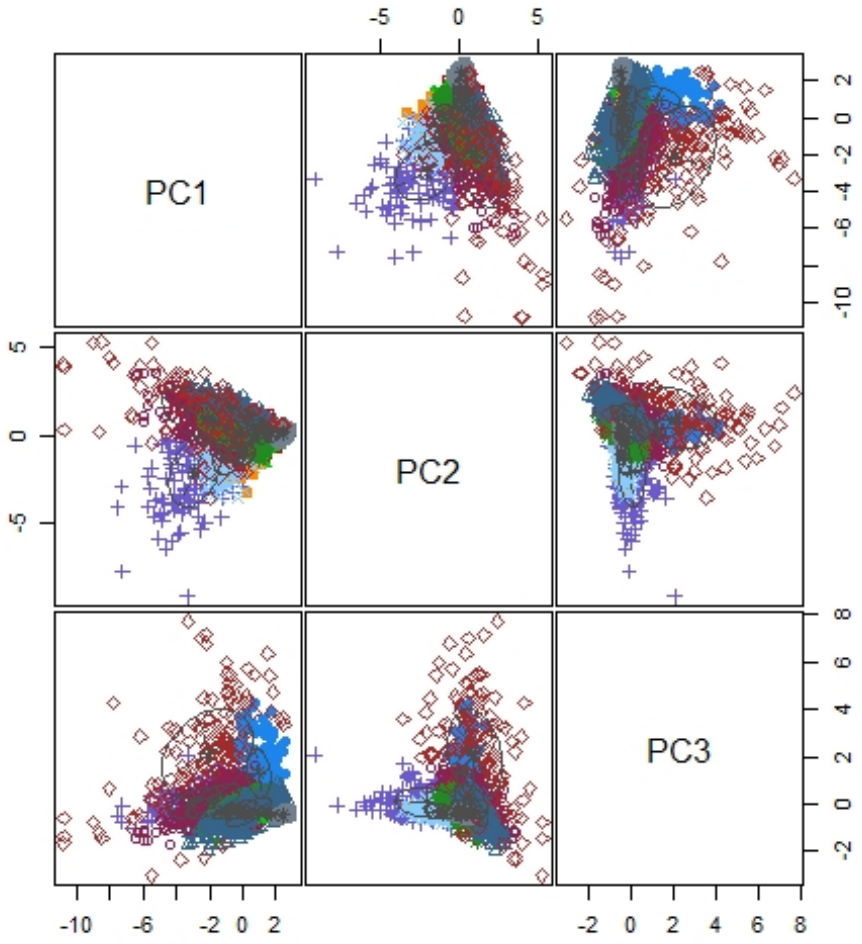

Figure 4.8: Gaussian Mixture Model - 11 components, Model type: VVE

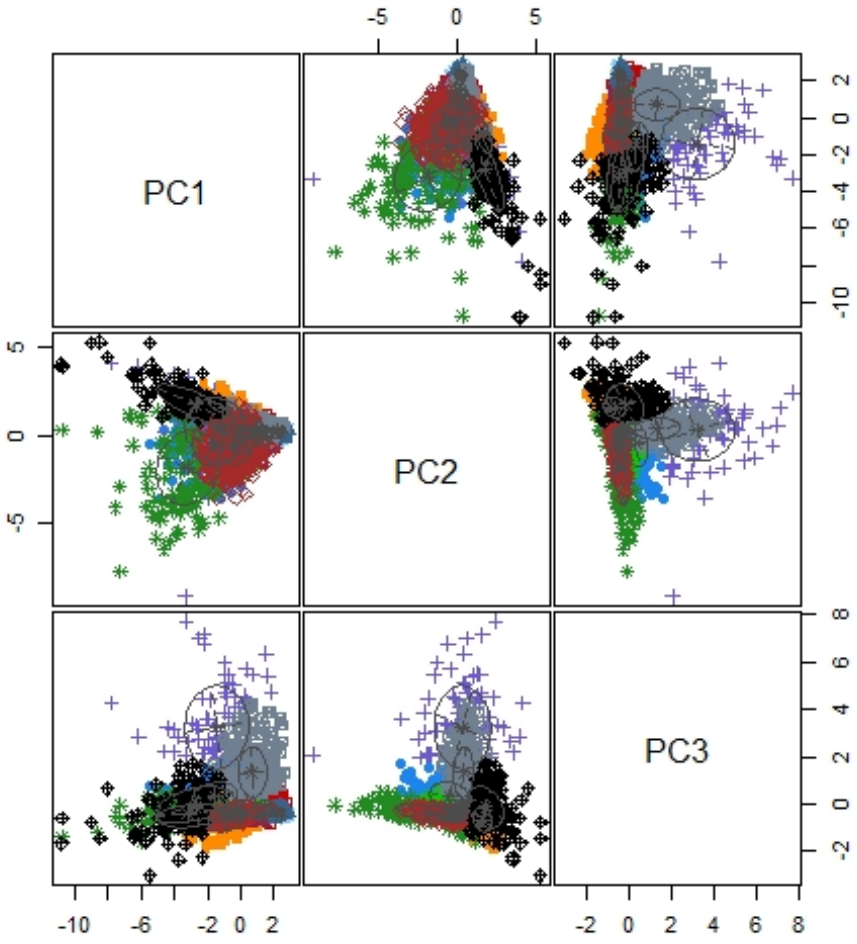

Figure 4.9: Gaussian Mixture Model - 12 components, Model type: VVE

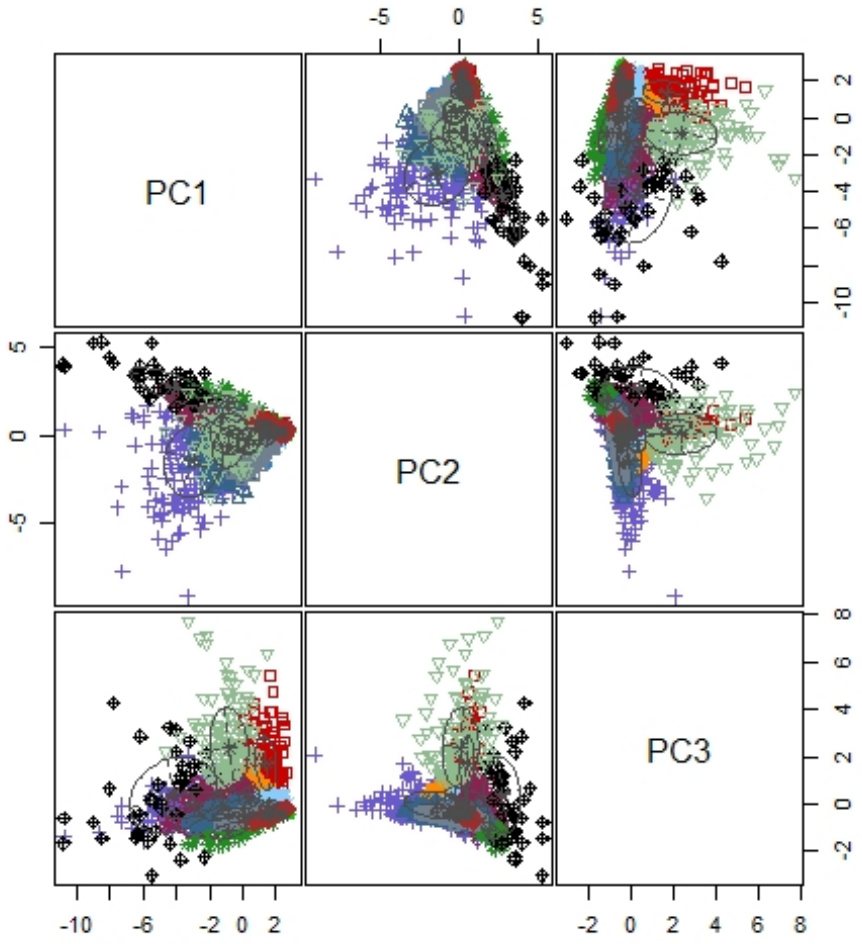

Figure 4.10: Gaussian Mixture Model - 13 components, Model type: VVE

4.8 Bernoulli Mixture Model

Instead of using the covariates in Table 4.1 to cluster casino patrons, as we have done up to this point, clustering can be accomplished in a different manner. Consider a matrix in which the rows represent patrons and the columns represent slot machines. If a player has used a slot machine in the time period, then a 1 is placed in the cell of that row and column. Conversely, if they have not used that slot machine, a 0 is placed in the cell. The resulting matrix is a binary matrix and can be visualized in the form of a heatmap shown in Figure 4.11. Each red block in the cell indicates a 1, while a yellow block indicates a 0.

To cluster patrons, ideally we would like to place patrons that have used similar slot machines into their own clusters. The Bernoulli mixture model is utilized to do so. First, we use the flexmix (Leisch, 2004) package to fit this model. As with the Gaussian mixture model, we initialize and run several iterations of the EM algorithm, each time keeping track of the optimal number of clusters based on the calculated BIC of the resulting model. As can be seen in Figure 4.12, the optimal number of clusters is most frequently 4 or 5. Therefore, we fit Bernoulli mixture model with both 4 and 5 components. Figure 4.13 to Figure 4.16 shows each cluster in a separate heatmap for the Bernoulli mixture model with 4 components. Figure 4.17 to Figure 4.21 shows each cluster for a Bernoulli mixture model with 5 components.

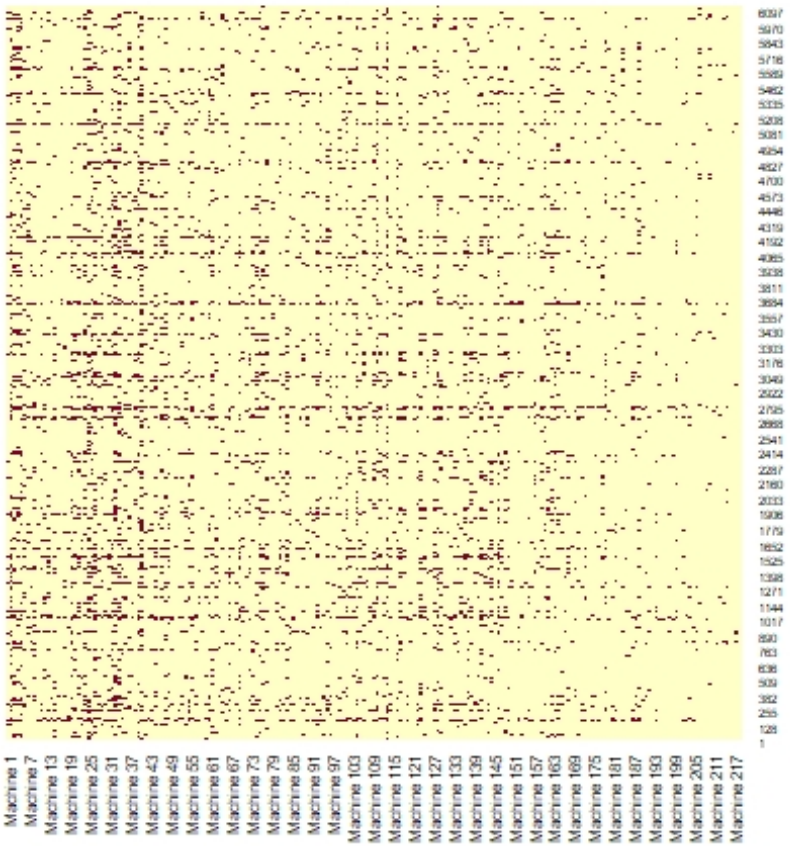

Figure 4.11: Heatmap of binary player matrix

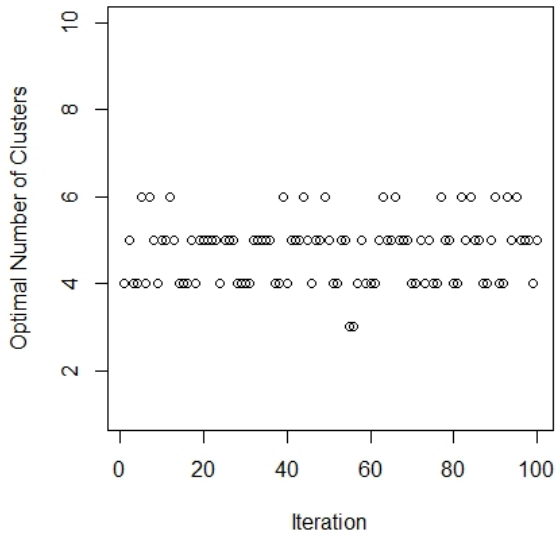

Figure 4.12: Optimal Number of Clusters for Bernoulli Mixture Model

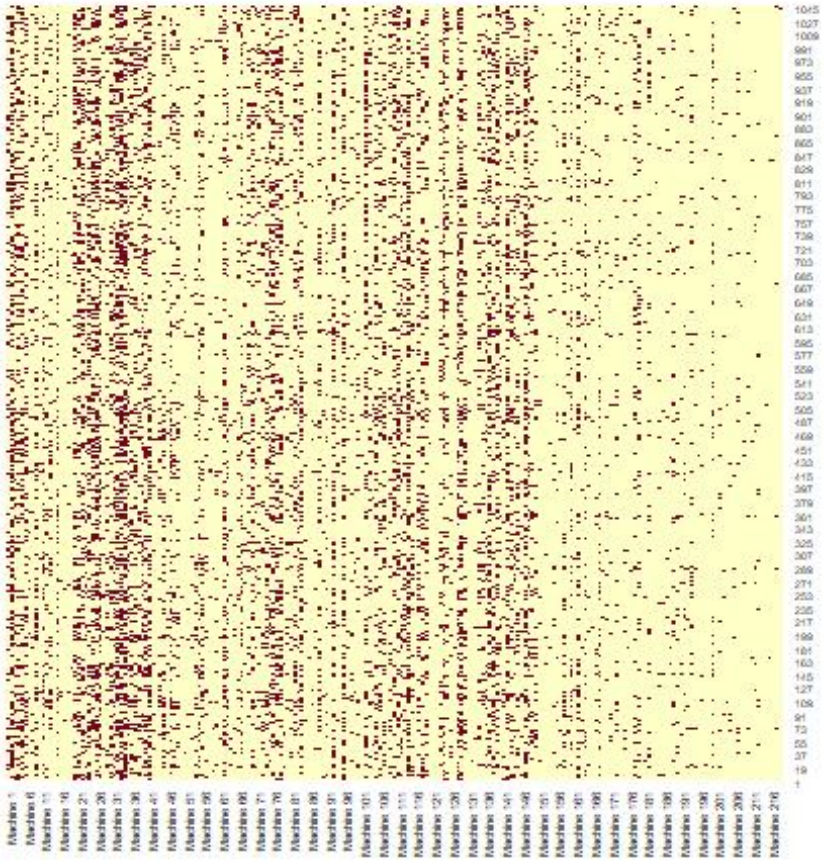

Figure 4.13: Cluster 1 of Bernoulli Mixture Model with 4 components

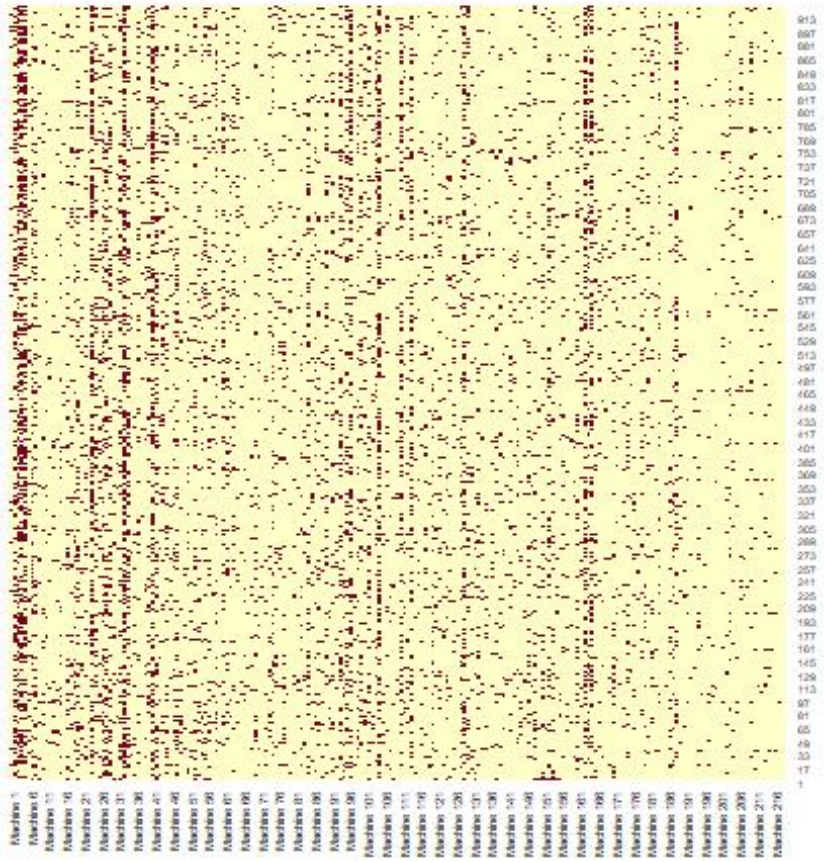

Figure 4.14: Cluster 2 of Bernoulli Mixture Model with 4 components

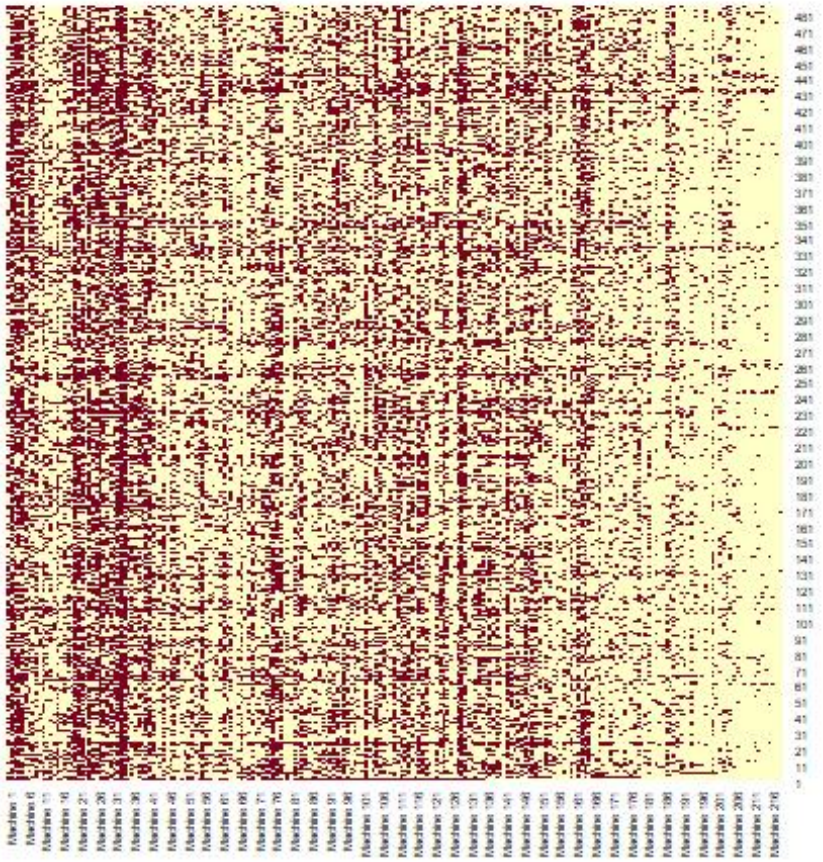

Figure 4.15: Cluster 3 of Bernoulli Mixture Model with 4 components

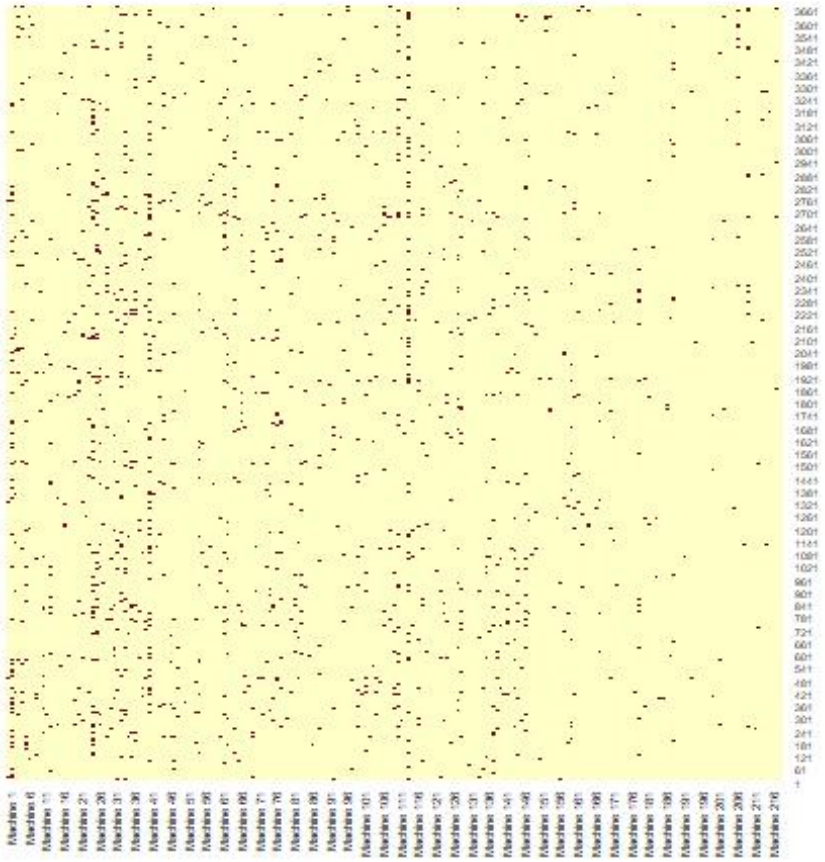

Figure 4.16: Cluster 4 of Bernoulli Mixture Model with 4 components

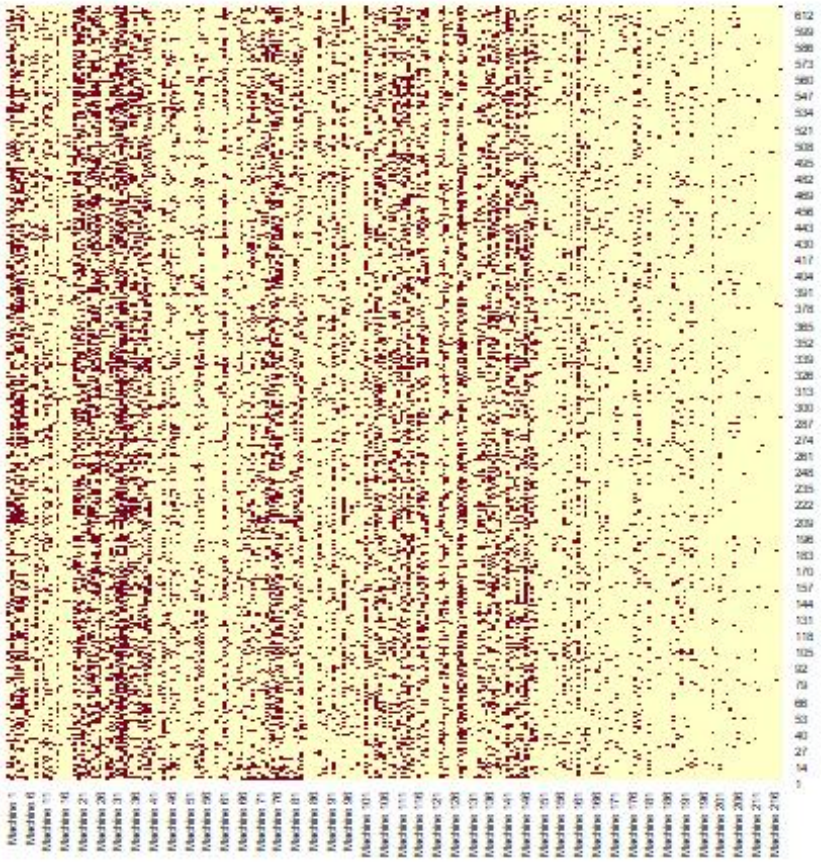

Figure 4.17: Cluster 1 of Bernoulli Mixture Model with 5 components

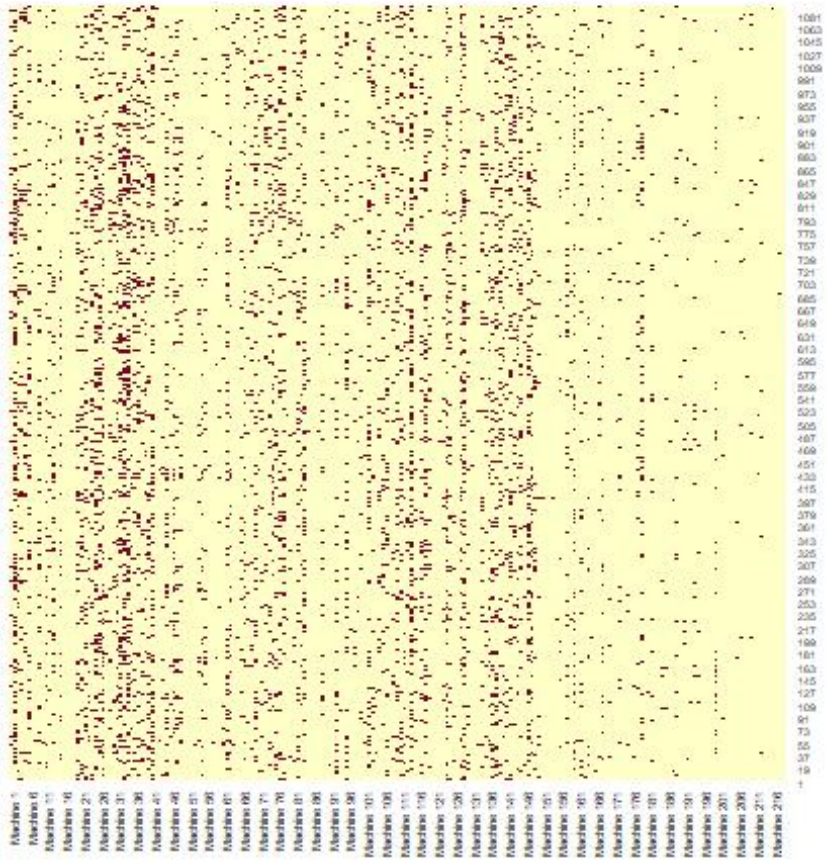

Figure 4.18: Cluster 2 of Bernoulli Mixture Model with 5 components

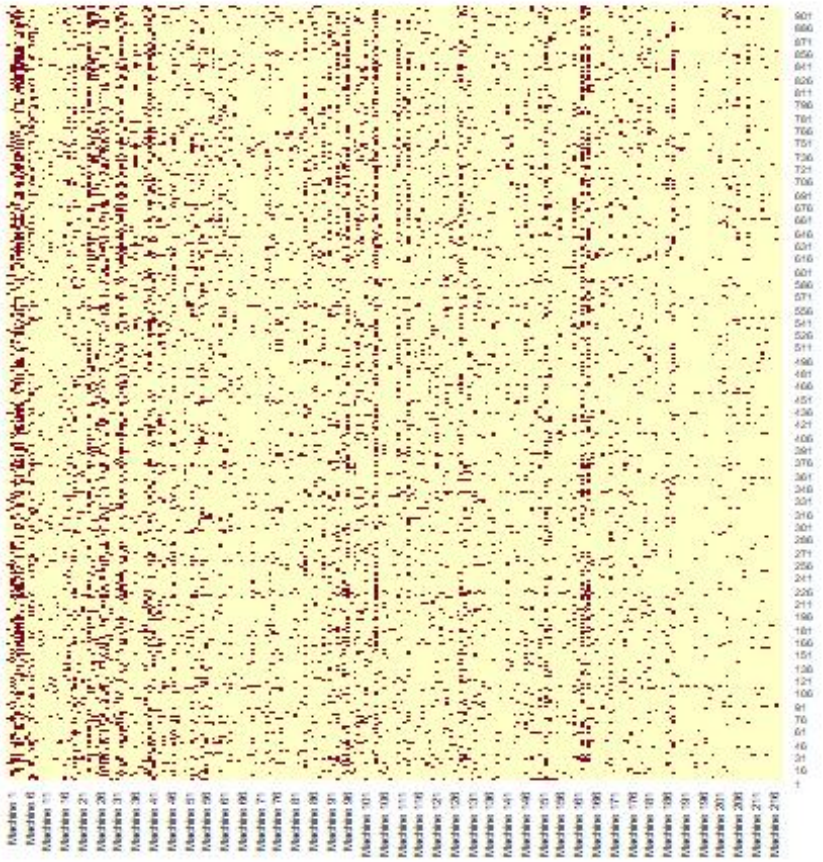

Figure 4.19: Cluster 3 of Bernoulli Mixture Model with 5 components

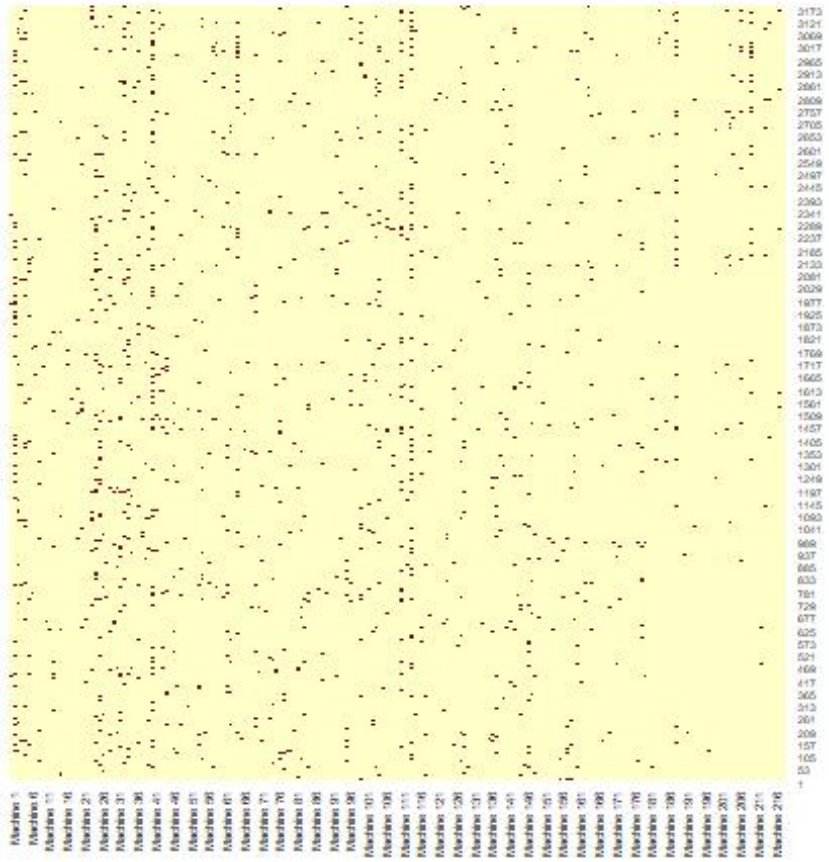

Figure 4.20: Cluster 4 of Bernoulli Mixture Model with 5 components

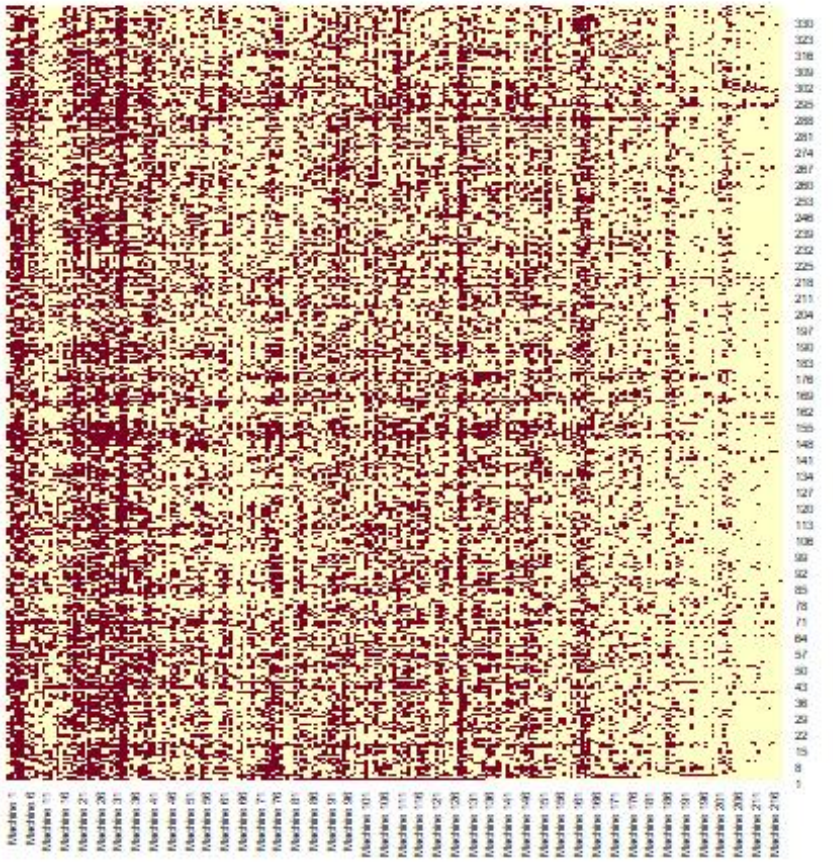

Figure 4.21: Cluster 5 of Bernoulli Mixture Model with 5 components

4.9 Interpretation of Clustering Results

In the previous sections, we have explored several candidate clustering partitions for the provided casino dataset. Determining the most suitable output among the candidates, however, is a challenging issue. For instance, despite excellent internal validation metric performance, a clustering output may have little to no practical meaning. As a result, choosing a clustering partition is often a combination of both statistical performance and domain-specific knowledge.

The above scenario is exemplified using the results from agglomerative hierarchical clustering in subsection 4.4.3. Despite excellent performance metrics when using two distinct clusters, the clustering algorithm placed nearly all patrons into one cluster, leaving only two patrons in the second cluster. This output resulted in some of the best performance metrics, but interpreting the results may be meaningless to stakeholders as the output does not adequately explain the structure of the dataset, and only serves to highlight the presence of two likely outliers in the data.

On the other hand, consider the output of the k-means algorithm using six clusters. Although the evaluation metrics may not be as favourable as those obtained from hierarchical clustering, the output is much more meaningful. The number of patrons placed into each cluster is shown in Table 4.40.

To interpret these clusters, we will gather the patrons placed into each cluster and examine the mean values of the original covariates. These values for each cluster is shown in Table 4.41.

Cluster	Number of Patrons
1	144
2	978
3	126
4	172
5	462
6	540

Table 4.40: Number of patrons placed into each cluster using k-means clustering (six clusters)

Cluster	total visits	mean duration	sd duration	mean games	sd games	mean machines	sd machines
1	111.02	3995.25	2768.32	812.12	533.93	4.07	2.56
2	16.56	2274.97	1652.22	389.19	268.95	2.56	1.48
3	32.89	14542.37	9369.55	2731.41	1475.61	5.18	3.13
4	22.08	7473.61	4243.76	1569.10	844.82	14.69	8.02
5	23.50	7426.01	5864.43	1343.95	812.96	3.74	2.23
6	19.26	4283.09	2675.18	851.69	520.36	6.71	4.07

Table 4.41: Mean values of the original covariates for the patrons placed into each cluster

Finally, these values are converted to z-scores by subtracting the column mean across the whole dataset and dividing by the standard deviation of that column. By transforming the values to z-scores, we are able to easily see how the cluster means relate to the original covariates. Note that positive values indicate that the cluster mean is higher than average, while negative values indicate that the cluster mean is lower than average. The z-scores of each cluster is shown in Table 4.42.

Using the z-scores, we can then apply meaning to each cluster. For example,

Cluster	total visits	mean duration	sd duration	mean games	sd games	mean machines	sd machines
1	3.03	-0.23	-0.18	-0.13	-0.04	-0.19	-0.12
2	-0.31	-0.70	-0.55	-0.70	-0.70	-0.57	-0.61
3	0.27	2.69	1.95	2.49	2.33	0.10	0.14
4	-0.12	0.73	0.29	0.91	0.75	2.53	2.37
5	-0.07	0.72	0.82	0.60	0.67	-0.27	-0.27
6	-0.21	-0.15	-0.21	-0.07	-0.07	0.49	0.57

Table 4.42: Z-scores of the original covariates for the patrons placed into each cluster

the patrons in the first cluster tend to have a much high number of visits (total_visits) compared to the other clusters. On the other hand, patrons in the third cluster tend to have a relatively average number of visits but tend to stay at the casino for a much longer duration and play more games. Patrons in cluster 4 tend to stay at the casino for a relatively average duration, but tend to use a large number of machines within their visit.

This strategy of converting cluster means to z-scores can be applied to the various clustering outputs we have explored in previous sections. By applying such interpretations to clustering partitions, stakeholders may find value and be able to use this knowledge to their advantage. It is worth noting that different clustering partitions will undoubtedly result in different interpretations and there is likely no single "best" answer. Therefore, choosing a clustering partition may become a somewhat subjective task. This issue is pervasive throughout the field of unsupervised learning.

The strategy discussed above is applicable to clustering methods using

quantitative data. When using binary data, however, z-scores become meaningless. Thus, a different method must be adopted to provide meaning to the clusters. We will now discuss an interpretation of the Bernoulli mixture model output given in section 4.8.

Recall that in the Bernoulli mixture model, we clustered patrons based only on the specific machines they used within their visits to the casino across the entire time period. To provide meaning to the clusters of the Bernoulli mixture model, we will use the parameters output by the model. Each parameter is denoted θ_{jd}, where j indicates a particular component (cluster), and d indicates a particular machine, with $0 \leq \theta_{jd} \leq 1$. Furthermore, θ_{jd} is interpreted as the probability that a patron from component j has played machine d. All θ values are given in Table 4.43. Furthermore, we can display these theta values in the form of a heatmap Figure 4.22.

	Comp.1	Comp.2	Comp.3	Comp.4
Machine 1	0.15	0.04	0.38	0.01
Machine 2	0.43	0.26	0.69	0.12
Machine 3	0.22	0.42	0.64	0.04
Machine 4	0.23	0.43	0.64	0.04
Machine 5	0.26	0.43	0.67	0.05
Machine 6	0.21	0.07	0.36	0.04
Machine 7	0.04	0.23	0.47	0.01
Machine 8	0.06	0.20	0.45	0.01
Machine 9	0.23	0.04	0.42	0.02
Machine 10	0.01	0.05	0.18	0.00
Machine 11	0.20	0.04	0.44	0.01
Machine 12	0.13	0.02	0.24	0.01

Continued on next page

	Comp.1	Comp.2	Comp.3	Comp.4
Machine 13	0.18	0.05	0.28	0.03
Machine 14	0.01	0.04	0.13	0.00
Machine 15	0.25	0.06	0.44	0.02
Machine 16	0.01	0.03	0.19	0.00
Machine 17	0.03	0.09	0.34	0.01
Machine 18	0.02	0.12	0.36	0.01
Machine 19	0.02	0.08	0.30	0.00
Machine 20	0.28	0.18	0.73	0.03
Machine 21	0.28	0.07	0.59	0.02
Machine 22	0.28	0.09	0.57	0.02
Machine 23	0.10	0.01	0.20	0.00
Machine 24	0.26	0.41	0.68	0.04
Machine 25	0.32	0.18	0.47	0.12
Machine 26	0.37	0.13	0.54	0.07
Machine 27	0.39	0.26	0.76	0.05
Machine 28	0.05	0.25	0.54	0.01
Machine 29	0.13	0.22	0.49	0.04
Machine 30	0.21	0.05	0.44	0.02
Machine 31	0.23	0.03	0.44	0.02
Machine 32	0.35	0.16	0.58	0.06
Machine 33	0.45	0.45	0.83	0.07
Machine 34	0.41	0.22	0.68	0.08
Machine 35	0.18	0.03	0.35	0.02
Machine 36	0.21	0.04	0.39	0.02
Machine 37	0.23	0.12	0.39	0.05
Machine 38	0.21	0.03	0.43	0.01
Machine 39	0.21	0.06	0.43	0.02
Machine 40	0.28	0.12	0.56	0.03
Machine 41	0.33	0.38	0.58	0.16
Machine 42	0.04	0.21	0.40	0.01
Machine 43	0.01	0.05	0.14	0.01

Continued on next page

	Comp.1	Comp.2	Comp.3	Comp.4
Machine 44	0.08	0.12	0.25	0.02
Machine 45	0.20	0.11	0.45	0.03
Machine 46	0.06	0.03	0.14	0.01
Machine 47	0.28	0.08	0.42	0.04
Machine 48	0.15	0.19	0.41	0.03
Machine 49	0.16	0.05	0.25	0.02
Machine 50	0.01	0.04	0.20	0.00
Machine 51	0.01	0.07	0.27	0.00
Machine 52	0.03	0.06	0.28	0.00
Machine 53	0.02	0.15	0.36	0.01
Machine 54	0.13	0.07	0.25	0.01
Machine 55	0.20	0.09	0.32	0.05
Machine 56	0.18	0.15	0.55	0.02
Machine 57	0.03	0.10	0.31	0.01
Machine 58	0.02	0.07	0.21	0.01
Machine 59	0.01	0.12	0.15	0.02
Machine 60	0.05	0.02	0.15	0.00
Machine 61	0.03	0.15	0.33	0.02
Machine 62	0.26	0.10	0.52	0.03
Machine 63	0.15	0.05	0.26	0.03
Machine 64	0.01	0.09	0.31	0.00
Machine 65	0.03	0.07	0.06	0.06
Machine 66	0.10	0.02	0.22	0.01
Machine 67	0.22	0.04	0.40	0.02
Machine 68	0.06	0.01	0.10	0.01
Machine 69	0.13	0.04	0.27	0.01
Machine 70	0.21	0.08	0.41	0.03
Machine 71	0.10	0.04	0.20	0.01
Machine 72	0.11	0.02	0.22	0.01
Machine 73	0.13	0.03	0.32	0.01
Machine 74	0.20	0.08	0.44	0.02

Continued on next page

	Comp.1	Comp.2	Comp.3	Comp.4
Machine 75	0.25	0.16	0.50	0.02
Machine 76	0.27	0.07	0.57	0.02
Machine 77	0.27	0.07	0.47	0.04
Machine 78	0.30	0.09	0.56	0.03
Machine 79	0.04	0.01	0.09	0.00
Machine 80	0.08	0.04	0.21	0.01
Machine 81	0.07	0.05	0.11	0.02
Machine 82	0.21	0.03	0.41	0.02
Machine 83	0.14	0.06	0.28	0.01
Machine 84	0.26	0.04	0.47	0.02
Machine 85	0.02	0.20	0.33	0.01
Machine 86	0.00	0.06	0.18	0.00
Machine 87	0.03	0.08	0.24	0.01
Machine 88	0.02	0.10	0.32	0.01
Machine 89	0.21	0.04	0.32	0.02
Machine 90	0.02	0.13	0.33	0.01
Machine 91	0.07	0.01	0.20	0.00
Machine 92	0.03	0.13	0.36	0.01
Machine 93	0.24	0.05	0.44	0.02
Machine 94	0.04	0.20	0.48	0.01
Machine 95	0.01	0.07	0.23	0.00
Machine 96	0.30	0.17	0.51	0.04
Machine 97	0.03	0.31	0.41	0.01
Machine 98	0.06	0.02	0.22	0.01
Machine 99	0.02	0.12	0.32	0.00
Machine 100	0.07	0.13	0.28	0.03
Machine 101	0.01	0.10	0.15	0.02
Machine 102	0.30	0.13	0.58	0.03
Machine 103	0.09	0.05	0.26	0.02
Machine 104	0.03	0.14	0.22	0.02
Machine 105	0.22	0.39	0.59	0.03

Continued on next page

	Comp.1	Comp.2	Comp.3	Comp.4
Machine 106	0.18	0.03	0.38	0.01
Machine 107	0.21	0.06	0.38	0.03
Machine 108	0.00	0.04	0.18	0.00
Machine 109	0.12	0.02	0.25	0.01
Machine 110	0.25	0.04	0.46	0.02
Machine 111	0.25	0.23	0.47	0.08
Machine 112	0.20	0.05	0.47	0.01
Machine 113	0.13	0.06	0.36	0.01
Machine 114	0.43	0.26	0.55	0.16
Machine 115	0.14	0.02	0.33	0.02
Machine 116	0.00	0.05	0.14	0.00
Machine 117	0.20	0.05	0.47	0.02
Machine 118	0.27	0.09	0.44	0.03
Machine 119	0.22	0.04	0.45	0.01
Machine 120	0.01	0.12	0.29	0.01
Machine 121	0.00	0.04	0.10	0.00
Machine 122	0.02	0.11	0.32	0.00
Machine 123	0.13	0.03	0.36	0.01
Machine 124	0.36	0.11	0.60	0.03
Machine 125	0.18	0.06	0.36	0.01
Machine 126	0.04	0.06	0.19	0.01
Machine 127	0.01	0.03	0.12	0.00
Machine 128	0.30	0.10	0.53	0.04
Machine 129	0.37	0.26	0.77	0.05
Machine 130	0.17	0.12	0.52	0.01
Machine 131	0.01	0.07	0.23	0.00
Machine 132	0.00	0.06	0.25	0.00
Machine 133	0.02	0.11	0.33	0.01
Machine 134	0.17	0.05	0.40	0.01
Machine 135	0.14	0.05	0.36	0.01
Machine 136	0.14	0.02	0.27	0.01

Continued on next page

	Comp.1	Comp.2	Comp.3	Comp.4
Machine 137	0.27	0.05	0.47	0.02
Machine 138	0.22	0.06	0.34	0.03
Machine 139	0.26	0.06	0.44	0.03
Machine 140	0.12	0.02	0.28	0.01
Machine 141	0.07	0.02	0.11	0.01
Machine 142	0.32	0.09	0.60	0.02
Machine 143	0.17	0.06	0.36	0.02
Machine 144	0.16	0.04	0.32	0.02
Machine 145	0.22	0.05	0.35	0.03
Machine 146	0.08	0.02	0.25	0.01
Machine 147	0.40	0.15	0.57	0.05
Machine 148	0.22	0.03	0.40	0.02
Machine 149	0.16	0.10	0.49	0.01
Machine 150	0.10	0.02	0.34	0.01
Machine 151	0.01	0.10	0.32	0.00
Machine 152	0.00	0.01	0.09	0.00
Machine 153	0.03	0.20	0.44	0.01
Machine 154	0.02	0.12	0.34	0.01
Machine 155	0.05	0.02	0.12	0.00
Machine 156	0.07	0.06	0.22	0.01
Machine 157	0.01	0.06	0.19	0.01
Machine 158	0.15	0.04	0.29	0.02
Machine 159	0.04	0.02	0.09	0.01
Machine 160	0.24	0.02	0.38	0.03
Machine 161	0.01	0.16	0.26	0.02
Machine 162	0.22	0.05	0.51	0.01
Machine 163	0.02	0.32	0.47	0.01
Machine 164	0.17	0.13	0.57	0.01
Machine 165	0.03	0.31	0.41	0.01
Machine 166	0.02	0.07	0.28	0.00
Machine 167	0.07	0.06	0.18	0.02

Continued on next page

	Comp.1	Comp.2	Comp.3	Comp.4
Machine 168	0.17	0.03	0.34	0.01
Machine 169	0.01	0.04	0.15	0.00
Machine 170	0.05	0.03	0.15	0.01
Machine 171	0.03	0.13	0.33	0.01
Machine 172	0.01	0.04	0.17	0.01
Machine 173	0.09	0.02	0.16	0.02
Machine 174	0.02	0.04	0.20	0.00
Machine 175	0.03	0.08	0.28	0.01
Machine 176	0.00	0.02	0.06	0.00
Machine 177	0.01	0.06	0.19	0.00
Machine 178	0.13	0.04	0.24	0.01
Machine 179	0.20	0.09	0.34	0.05
Machine 180	0.01	0.06	0.13	0.00
Machine 181	0.01	0.08	0.16	0.01
Machine 182	0.11	0.07	0.29	0.01
Machine 183	0.00	0.02	0.04	0.00
Machine 184	0.00	0.12	0.17	0.01
Machine 185	0.04	0.01	0.08	0.00
Machine 186	0.02	0.01	0.08	0.00
Machine 187	0.02	0.10	0.35	0.00
Machine 188	0.01	0.07	0.30	0.00
Machine 189	0.08	0.20	0.30	0.05
Machine 190	0.05	0.04	0.17	0.00
Machine 191	0.05	0.02	0.16	0.00
Machine 192	0.05	0.01	0.12	0.00
Machine 193	0.02	0.01	0.05	0.00
Machine 194	0.10	0.02	0.24	0.01
Machine 195	0.02	0.04	0.13	0.00
Machine 196	0.00	0.01	0.04	0.00
Machine 197	0.02	0.01	0.04	0.00
Machine 198	0.00	0.02	0.11	0.00

Continued on next page

	Comp.1	Comp.2	Comp.3	Comp.4
Machine 199	0.00	0.01	0.02	0.00
Machine 200	0.16	0.02	0.32	0.02
Machine 201	0.00	0.00	0.02	0.00
Machine 202	0.04	0.04	0.15	0.00
Machine 203	0.01	0.06	0.21	0.00
Machine 204	0.01	0.10	0.26	0.01
Machine 205	0.01	0.04	0.18	0.01
Machine 206	0.02	0.02	0.08	0.00
Machine 207	0.04	0.07	0.09	0.03
Machine 208	0.00	0.01	0.02	0.00
Machine 209	0.01	0.03	0.06	0.00
Machine 210	0.02	0.09	0.04	0.04
Machine 211	0.00	0.03	0.05	0.00
Machine 212	0.02	0.02	0.08	0.00
Machine 213	0.03	0.01	0.06	0.00
Machine 214	0.00	0.05	0.07	0.01
Machine 215	0.00	0.03	0.06	0.00
Machine 216	0.00	0.01	0.03	0.00
Machine 217	0.00	0.01	0.03	0.00
Machine 218	0.01	0.02	0.02	0.01
Machine 219	0.00	0.00	0.01	0.00

Table 4.43: Theta values for the Bernoulli mixture model with 4 components

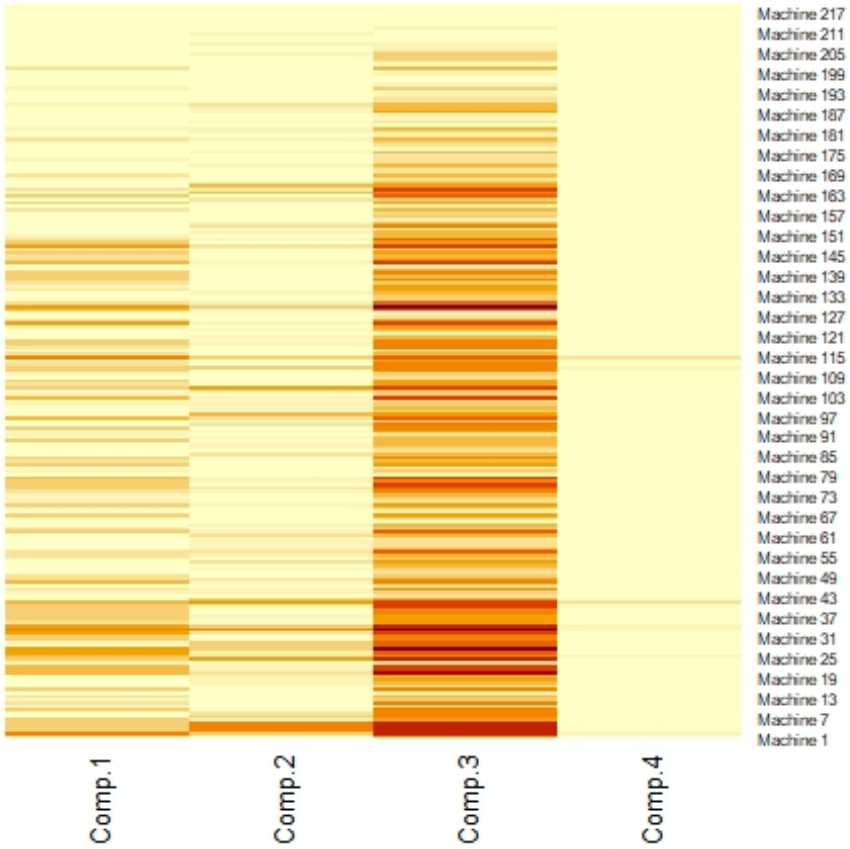

Figure 4.22: Heatmap of θ values for Bernoulli mixture model with 4 components

From the heatmap and table provided, we can immediately make a few important observations. Firstly, patrons from the fourth component are unlikely to play any machines. On the other hand, patrons from the third component are extremely likely to play a large number of machines. The first and second component offer a middle ground between the other two components. These

findings are in agreement with the cluster heatmaps shown in section 4.8.

With auxiliary information available about the slot machines themselves, further analysis can be performed. For instance, we can obtain the top five slot machines that a patron from each cluster is most likely to play. Then, we can examine any commonalities between these machines to investigate what qualities patrons from each component is attracted to in a slot machine. Using the second cluster as an example, it was found that the top five machines most likely to be played (i.e. with the highest θ value) had the same theoretical hold, the same manufacturer, and were essentially the same game. Similarly, the same can be done for the machines least likely to be played. Using auxiliary information to aid in the interpretation of clusters is an important tool and can ultimately reveal important characteristics of each cluster.

4.10 Discussion

By applying the clustering methods discussed in Chapter 2 to casino data, a wide variety of clustering outputs was found. Furthermore, by applying internal evaluation metrics to the resulting clusters, we were able to examine which cluster outputs were valid from a statistical standpoint. In particular, agglomerative hierarchical clustering using single-linkage with two clusters performed best in terms of the calculated metrics. On the other hand, these clusters did not have much practical meaning due to a large imbalance in the number of patrons placed in each cluster. Instead, a combination of domain-specific knowledge, evaluation metrics, and an appropriate clustering algorithm

for the data is necessary to choose one of the candidate outputs. Consequently, choosing a single best output is extremely difficult and sometimes impossible without prior class labels provided, which is rarely provided. Rather, the best clustering output is often the one that makes sense to stakeholders and provides meaningful clusters.

On that note, one of the clustering methods that stood out to our industry partner was the Gaussian mixture model due to its advantages over other methods. These advantages include being able to specify the shapes of the clusters through constraints on the covariance matrix, providing an underlying model to the clustering rather than just clustering based on a heuristic, and being able to use the Bayesian Information Criterion to suggest the appropriate number of clusters. Based on the evaluation metrics of the six different Gaussian mixture models, the two best models were the "VVV" model with 11 clusters or the "VVE" model with 13 clusters. The number of patrons placed into each cluster is given in Table 4.44 and Table 4.47.

In the previous section, we discussed a strategy for interpreting clusters based on converting the cluster means to z-scores; these scores for the two Gaussian mixture models are given in Table 4.45 and Table 4.48.

We can also calculate the stability metrics discussed in section 3.4 for both of these clustering partitions. The cluster-wise mean Jaccard similarity coefficients for each GMM model are given in Table 4.46 and Table 4.49 for 100 bootstrap iterations. As a reminder, values close to 1 indicate that the clustering partitions are stable, while values close to 0 indicate unstable clusters.

The cluster-wise similarity coefficients indicate that some of the clusters in each model may be unstable, while others are relatively stable. In other words, given bootstrapped samples, some of the clusters in the original partition were able to be consistently recovered, while others were not.

The Bernoulli mixture model applied to the binary data also found meaningful clusters. The visualization of the clusters in the form of heatmaps is especially convincing regarding the validity of the output. However, due to the large number of parameters output by the model, interpretation of these models may be slightly more difficult. Further investigation into both candidate Bernoulli mixture models is needed to determine valid interpretations and a selection among them.

Cluster	Number of Patrons
1	71
2	132
3	378
4	329
5	181
6	207
7	172
8	404
9	269
10	89
11	190

Table 4.44: Number of patrons placed into each cluster of the Gaussian mixture model (VVV) with 11 components

Cluster	total visits	mean duration	sd duration	mean games	sd games	mean machines	sd machines
1	0.29	2.83	1.89	2.75	2.66	0.21	0.30
2	-0.03	0.96	0.48	1.13	0.87	2.70	2.42
3	-0.62	-0.41	-0.37	-0.43	-0.42	-0.33	-0.35
4	-0.22	-0.69	-0.53	-0.68	-0.65	-0.58	-0.59
5	1.67	-0.46	-0.34	-0.41	-0.36	-0.36	-0.27
6	-0.59	-1.02	-0.82	-1.01	-1.09	-0.78	-0.90
7	0.11	-0.44	-0.43	-0.36	-0.38	0.32	0.36
8	-0.51	0.08	-0.07	0.14	0.12	0.61	0.69
9	-0.18	0.93	1.20	0.69	0.73	-0.10	-0.05
10	3.33	0.59	0.31	0.74	0.77	0.22	0.24
11	0.27	0.48	0.52	0.41	0.60	-0.66	-0.66

Table 4.45: Z-scores of the original covariates for each cluster of the Gaussian mixture model (VVV) with 11 components

Cluster	Mean Jaccard Similarity Coefficient
1	0.5075
2	0.5394
3	0.3909
4	0.5209
5	0.5760
6	0.5411
7	0.4763
8	0.5123
9	0.4882
10	0.3077
11	0.4783

Table 4.46: Cluster-wise mean Jaccard coefficient for Gaussian mixture model (VVV) with 11 components (100 iterations)

Cluster	Number of Patrons
1	200
2	106
3	360
4	261
5	119
6	213
7	154
8	246
9	148
10	141
11	187
12	118
13	169

Table 4.47: Number of patrons placed into each cluster of the Gaussian mixture model (VVE) with 13 components

Cluster	total visits	mean duration	sd duration	mean games	sd games	mean machines	sd machines
1	1.16	-0.47	-0.29	-0.44	-0.36	-0.44	-0.38
2	-0.57	-0.54	-0.51	-0.49	-0.51	0.40	0.52
3	-0.07	0.35	0.21	0.40	0.37	0.15	0.18
4	-0.13	-0.68	-0.54	-0.67	-0.66	-0.57	-0.59
5	-0.40	0.83	1.84	0.17	0.44	-0.71	-0.77
6	-0.61	-0.68	-0.57	-0.68	-0.68	-0.37	-0.34
7	-0.63	-1.05	-0.85	-1.04	-1.13	-0.78	-0.92
8	-0.49	0.11	-0.01	0.17	0.20	0.81	0.96
9	-0.52	-0.68	-0.49	-0.76	-0.75	-0.80	-0.91
10	0.45	1.97	1.36	1.93	1.90	-0.29	-0.24
11	-0.57	-0.00	-0.07	0.01	0.02	-0.18	-0.22
12	3.32	0.22	0.11	0.35	0.44	0.13	0.19
13	0.05	1.08	0.55	1.25	1.02	2.37	2.16

Table 4.48: Z-scores of the original covariates for each cluster of the Gaussian mixture model (VVE) with 13 components

Cluster	Mean Jaccard Similarity Coefficient
1	0.4051
2	0.6497
3	0.7377
4	0.5285
5	0.5754
6	0.6674
7	0.4601
8	0.5739
9	0.2407
10	0.4727
11	0.3659
12	0.3604
13	0.5855

Table 4.49: Cluster-wise mean Jaccard coefficient for Gaussian mixture model (VVE) with 13 components (100 iterations)

Chapter 5

Conclusion

Several clustering methods, evaluation metrics, and strategies for interpretation were explored throughout this work. By applying these practices to a dataset obtained from a casino, we were able to find potential clusters of patrons that had visited that casino in a certain time period. In particular, the clusters selected were found through model-based clustering (Gaussian mixture models and Bernoulli mixture models). These clusters enabled us to find structure within the dataset and learn new information about how these patrons behaved and how they used the slot machines within the casino. We hope that the practices discussed will generalize to data from other casinos so that others are also able to find meaningful clusters among their patrons.

Furthermore, despite growth in clustering research over the past several years, the clustering of casino patrons is not well-studied and there is a lack of literature in this area. This may be due to the confidentiality of casino data and the competitive nature of the casino industry. We hope that this work will

encourage casinos and researchers to collaborate so that further research into gambling behaviour can occur. Clustering casino patrons may be used in the future for purposes such as identifying problematic gambling behaviour. As a result, research in this area is of great importance.

Bibliography

Akaike, H. (1974). A New Look at the Statistical Model Identification. *IEEE Transactions on Automatic Control*, 19(6):716–723. (Cited on page 23.)

American Gaming Association (2019). Gaming by the numbers - gaming's national impact. (Cited on page 1.)

Ben-David, S. and Luxburg, U. (2008). Relating clustering stability to properties of cluster boundaries. pages 379–390. (Cited on page 66.)

Bezdek, J. C. (1981). Models for Pattern Recognition. *Pattern Recognition with Fuzzy Objective Function Algorithms*, pages 1–13. (Cited on page 21.)

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg. (Cited on pages 24, 29, 45 and 47.)

Brock, G., Pihur, V., Datta, S., and Datta, S. (2008). clValid: An R package for cluster validation. *Journal of Statistical Software*, 25(4):1–22. (Cited on page 59.)

Cattell, R. B. (1966). The scree test for the number of factors. *Multivariate Behavioral Research*, 1(2):245–276. PMID: 26828106. (Cited on page 10.)

Cebeci, Z. (2019). Comparison of internal validity indices for fuzzy clustering. *Journal of Agricultural Informatics*, pages 1–14. (Cited on page 22.)

Chen, S. C., Shoemaker, S., and Zemke, D. M. V. (2013). Segmenting slot machine players: A factor-cluster analysis. *International Journal of Contemporary Hospitality Management*, 25(1):23–48. (Cited on page 4.)

Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227. (Cited on page 57.)

Delua, J. (2021). Supervised vs. Unsupervised Learning: What's the Difference? — IBM. (Cited on page 3.)

Dempster, A. P., Laird, . N. M., and Rubin, . D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38. (Cited on page 27.)

Dunn, J. C. (1974). Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1):95–104. (Cited on page 55.)

Dunn, J. C. (2008). A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *https://doi.org/10.1080/01969727308546046*, 3(3):32–57. (Cited on page 19.)

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press. (Cited on page 48.)

Farajian, M. A. and Mohammadi, S. (2010). Mining the banking customer behavior using clustering and association rules methods. *Int. J. Indust. Eng. Prod. Res*, 21:239–245. (Cited on page 3.)

F.R.S., K. P. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572. (Cited on page 6.)

Fry, C. and Manna, S. (2016). Can we group similar amazon reviews: A case study with different clustering algorithms. In *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)*, pages 374–377. (Cited on page 3.)

Hahsler, M., Piekenbrock, M., and Doran, D. (2019). dbscan: Fast density-based clustering with R. *Journal of Statistical Software*, 91(1):1–30. (Cited on page 51.)

Handl, J., Knowles, J., and Kell, D. B. (2005). Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21(15):3201–3212. (Cited on page 59.)

Hennig, C. (2007). Cluster-wise assessment of cluster stability. *Computational Statistics and Data Analysis*, 52(1):258–271. (Cited on pages 66, 67 and 68.)

Hennig, C. (2020). *fpc: Flexible Procedures for Clustering*. R package version 2.2-8. (Cited on page 70.)

Hopkins, B. and Skellam, J. (1954). A new method for determining the type of distribution of plant individuals. *Annals of Botany*, 18(70):213–227. (Cited on page 52.)

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2(1):193–218. (Cited on page 62.)

Iaci, R. and Singh, A. (2012). Clustering high dimensional sparse casino player tracking datasets clustering high dimensional sparse casino player tracking datasets ross iaci 1. *UNLV Gaming Research & Review Journal*. (Cited on page 4.)

Jaccard (1901). Distribution de la florine alpine dans la bassin de dranses et dans quelques regiones voisines. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:241–272. (Cited on page 69.)

Kassambara, A. and Mundt, F. (2020). *factoextra: Extract and Visualize the Results of Multivariate Data Analyses*. R package version 1.0.7. (Cited on page 54.)

Kaufman, L. and Rousseeuw, P. J. (1990). *Finding Groups in Data*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, USA. (Cited on pages 13, 14, 15 and 16.)

Krause, E. F. (1973). Taxicab Geometry. *The Mathematics Teacher*, 66(8):695–706. (Cited on page 13.)

Kreer, J. (1957). A question of terminology. *IRE Transactions on Information Theory*, 3(3):208–208. (Cited on page 64.)

Lee, C. K., Lee, Y. K., Bernhard, B. J., and Yoon, Y. S. (2006). Segmenting casino gamblers by motivation: A cluster analysis of Korean gamblers. *Tourism Management*, 27(5):856–866. (Cited on page 4.)

Leisch, F. (2004). FlexMix: A general framework for finite mixture models and latent class regression in R. *Journal of Statistical Software*, 11(8):1–18. (Cited on pages 47 and 101.)

Lloyd, S. P. (1982). Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137. (Cited on page 11.)

Macqueen, J. (1967). Some methods for classification and analysis of multivariate observations. (Cited on page 10.)

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., and Hornik, K. (2019). *cluster: Cluster Analysis Basics and Extensions*. R package version 2.1.0 —

For new features, see the 'Changelog' file (in the package source). (Cited on pages 14, 16 and 18.)

McLachlan, G. J. and Peel, D. (2000). *Finite mixture models*. Wiley Series in Probability and Statistics, New York. (Cited on page 23.)

Palacio-Niño, J.-O. and Galiano, F. (2019). Evaluation metrics for unsupervised learning algorithms. *ArXiv*, abs/1905.05667. (Cited on page 60.)

Papastamoulis, P. and Rattray, M. (2017). BayesBinMix: an R Package for Model Based Clustering of Multivariate Binary Data. *The R Journal*, 9(1):403–420. (Cited on page 47.)

Phillips, J., Tandoh, M., Noble, S., and Bush, V. (2004). The Value of Relationship Strength in Segmenting Casino Patrons. *Journal of Interactive Advertising*, 5(1):60–73. (Cited on page 4.)

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. (Cited on page 12.)

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850. (Cited on page 60.)

Romano, S., Xuan Vinh, N., Bailey, J., and Verspoor, K. (2016). Adjusting for Chance Clustering Comparison Measures. *Journal of Machine Learning Research*, 17:1–32. (Cited on page 63.)

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(C):53–65. (Cited on page 56.)

Sander, J., Ester, M., Kriegel, H.-P., and Xu, X. (1998). Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications. *Data Mining and Knowledge Discovery 1998 2:2*, 2(2):169–194. (Cited on page 50.)

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., and Xu, X. (2017). Dbscan revisited, revisited: Why and how you should (still) use dbscan. *ACM Trans. Database Syst.*, 42(3). (Cited on page 50.)

Schwartz, D. (2017). The average big las vegas strip casino, 2017. (Cited on page 1.)

Schwartz, D. (2018). How casinos use math to make money when you play the slots. *Forbes*. (Cited on page 2.)

Schwarz, G. (1978). "Estimating the Dimension of a Model.". *The Annals of Statistics*, 6(2):461 – 464. (Cited on page 23.)

Scrucca, L., Fop, M., Murphy, T. B., and Raftery, A. E. (2016a). mclust 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models. *The R journal*, 8(1):289. (Cited on pages 24 and 29.)

Scrucca, L., Fop, M., Murphy, T. B., and Raftery, A. E. (2016b). mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, 8(1):289–317. (Cited on page 30.)

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423. (Cited on pages 63 and 64.)

Wikipedia contributors (2021). Rand index — Wikipedia, the free encyclopedia. [Online; accessed 14-July-2021]. (Cited on page 62.)

Wu, J. and Lin, Z. (2005). Research on customer segmentation model by clustering. In *Proceedings of the 7th International Conference on Electronic Commerce*, pages 316–318. (Cited on page 3.)

Xuan Vinh, N., Epps, J., and Bailey, J. (2010). Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. *Journal of Machine Learning Research*, 11:2837–2854. (Cited on pages 65 and 66.)

YiLan, L. and RuTong, Z. (2015). *clustertend: Check the Clustering Tendency*. R package version 1.4. (Cited on page 54.)