การเปรียบเทียบระหว่างข้อมูลปกติและข้อมูลที่ผ่าน PCA โดยใช้ชุดข้อมูล Credit Card Customer Data ด้วยเทคนิค Clustering จากโปรแกรม R

จัดทำโดย

นางสาววนิศรา จงใจ 665020059-1

เสนอ

อ.ดร.เปรม จันทร์สว่าง

รายงานนี้เป็นส่วนหนึ่งของรายวิชา Multivariate

ปีการศึกษา 1/2566

มหาวิทยาลัยขอนแก่น

คำนำ

รายงานเล่มนี้เป็นส่วนหนึ่งของรายวิชา SC637703 Multivariate Analysis เพื่อให้ได้ความรู้ในเรื่องการทำข้อมูลด้วยวิธี Principal Component Analysis ควบคู่กับการจัดกลุ่ม Clustering ภายใต้ข้อมูลสาธารณะและได้ศึกษาเพื่อเป็นประโยชน์ต่อการเรียนรู้ในอนาคต

ทั้งนี้ผู้จัดทำหวังว่ารายงานเล่มนี้จะเป็นประโยชน์กับผู้อ่านไม่มากก็น้อยหากมีข้อเสนอแนะหรือเกิดข้อผิดพลาดประการใดขออภัยมา ณ ที่นี้ด้วย

นางสาววนิศรา จงใจ

สารบัญ

1. Principal Component Analysis

PCA (Principal Component Analysis) คือเทคนิคการวิเคราะห์ข้อมูลแบบทางสถิติที่ใช้ในการลดมิติของข้อมูลโดยรวมถึงการค้นหาค่าหลักหลาย ๆ (principal components) ที่สามารถแสดงถึงความแปรปรวน (variance) ในข้อมูลอย่างได้อย่างมาก โดยลดมิติเหล่านั้นลงให้น้อยลงที่สุด โดยที่ยังรักษาข้อมูลเฉยๆ หรือสูญเสียข้อมูลน้อยที่สุดเท่าที่จะเป็นไปได้

เราสามารถอธิบาย PCA ได้ดังนี้:

1.1 การลดมิติข้อมูล: PCA ช่วยลดมิติของข้อมูลที่มีมากมายให้น้อยลงโดยการค้นหาและเลือกเฉพาะค่าหลักที่สำคัญ (principal components) ที่มีผลสำคัญในการอธิบายความแปรปรวนในข้อมูล

1.2 การสร้างค่าใหม่: หลังจากที่ข้อมูลถูกคำนวณแล้ว, PCA สามารถใช้เวกเตอร์ข้อมูลเดิมและค่าหลักเหล่านั้นเพื่อสร้างเวกเตอร์ใหม่ที่เป็นผลลัพธ์จากการลดมิติ. เวกเตอร์ใหม่นี้เรียกว่า "ค่าเชิงเป้าหมาย" (scores) และสามารถนำมาใช้ในการแสดงข้อมูลในมิติใหม่ที่น้อยลง โดยยังรักษาข้อมูลสำคัญ

2. ขั้นตอนของการทำ PCA

1. ก่อนที่เราจะทำ PCA, ควรตรวจสอบข้อมูลเพื่อให้แน่ใจว่ามีความเหมาะสมสำหรับการใช้ PCA และข้อมูลไม่มีปัญหาเช่น Missing Value เป็นต้น

2. ในครั้ง, เราอาจต้องมาตรวจสอบข้อมูลเพิ่มเติมและดำเนินการเตรียมข้อมูล ด้วยวิธีการ Standardize data เพื่อทำให้ปรับค่าเฉลี่ยและความแปรปรวน

3. คำนวณเมทริกซ์สหสัมพันธ์ (covariance matrix) ของข้อมูล ซึ่งจะแสดงความสัมพันธ์ระหว่างตัวแปร. เมทริกซ์นี้มีขนาด n x n, โดย n คือจำนวนตัวแปรในข้อมูล

4. คำนวณค่า eigen (eigenvalues and eigenvectors) ของเมทริกซ์สหสัมพันธ์ eigenvalues and eigenvectors จะแสดงถึงความสัมพันธ์และความแปรปรวนในข้อมูล

5. เรียงลำดับค่าหลักเราจะเรียงลำดับค่าหลักตามค่าหลักจากมากไปน้อย เพื่อเลือกเฉพาะค่าหลักที่มีผลในการอธิบายความแปรปรวนสูงสุด

6. เลือก PC หลักเลือกเฉพาะค่าหลักที่เราต้องการใช้ในการลดมิติ จำนวนค่าหลักที่เราเลือกจะกำหนดจำนวนมิติใหม่ที่เราจะสร้างโดยในที่นี้จะใช้ 80% สำหรับการเก็บไว้

ทั้งนี้ในงานของเราจะทำรวมกับวิธีการ Clustering หรือการจัดกลุ่มเพื่อดูว่าข้อมูลกรณีไม่ได้ทำ PCA และกรณีข้อมูลที่ทำ PCA แตกต่างกันอย่างไร

3. Clustering

Clustering เป็นเทคนิคในการวิเคราะห์ข้อมูลที่มีจุดประสงค์ในการจัดกลุ่มข้อมูลที่คล้ายกันเข้าด้วยกันในกลุ่มหรือกลุ่มย่อยต่าง ๆ โดยไม่ต้องให้ข้อมูลใด ๆ ล่วงล้ำกว่าการตัดสินใจของคลัสเตอร์ กระบวนการที่มีจุดหมายเรียกประกอบกันของข้อมูลที่คล้ายคลึงกัน และเอามาจัดกลุ่มให้อยู่รวมกันในและกลุ่มต่าง ๆ โดยใช้ความคล้ายคลึงในลักษณะหลาย ๆ รูปแบบ เช่น ระยะทาง (Distance), คุณสมบัติ เป็นต้น

ทั้งนี้ในการทำ Clustering จะประกอบไปด้วย 2 ส่วนหลักในการทำครั้งได้แก่

3.1 ระยะห่าง (Distance)

ระยะห่างหรือ Distance คือการวัดระยะทางหรือความห่างระหว่างจุดสองในพื้นที่หรือขอบเขตที่กำหนดไว้เพื่อประเมินความคล้ายคลึงหรือความแตกต่างระหว่างข้อมูลหรือจุดข้อมูลที่กำลังถูกจัดกลุ่มหรือวิเคราะห์ การเลือกหรือกำหนดวิธีวัดระยะห่างสำคัญในกระบวนการ Clustering เนื่องจากมีผลต่อผลลัพธ์ที่ได้และความคล้ายคลึงของกลุ่มที่เราสร้างขึ้นในครั้งนี้ได้นำเสนอทั้งหมด 2 ตัวได้แก่

3.1.1 ระยะทางยูคลิเดียน (Euclidean Distance):

วิธีนี้ใช้ระยะทางเป็นแนวทางในการวัดความห่างระหว่างจุดสองในพื้นที่หรือขอบเขตสองมิติหรือมากกว่า. สำหรับจุด A และจุด B ในระบบพิกัด (x, y) ระยะทางยูคลิเดียนคือค่าระยะทางที่ใช้เรทต่างกันของค่า x และค่า y ของ A และ B ในสมการสามเหลี่ยม.

3.1.2 ระยะทางแมนฮัตตัน (Manhattan Distance): วิธีนี้วัดระยะทางเป็นผลรวมของความแตกต่างในค่า x และค่า y ระหว่างจุด A และ B ในระบบพิกัด (x, y). นั่นคือ การเดินในเมืองแบบสี่ทิศทางเท่ากับระยะทางแมนฮัตตัน.

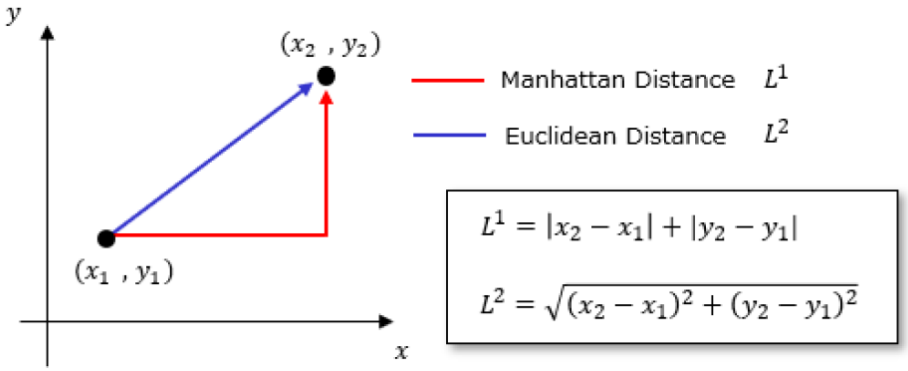

Figure 1 ตัวอย่างระยะห่างและวิธีคำนวณทั้ง 2 ตัว

3.2 ประเภทการ Clustering

ประเภทการจัดกลุ่มคือวิธีการการจัดกลุ่มโดยใช้ระยะห่างเป็นการจัดทั้งนี้ประเภทการจัดกลุ่มในครั้งนี้เราขอเสนอทั้งหมด 3 ประเภทได้แก่

3.1.1 K-Means เป็นวิธีการในการจัดกลุ่มข้อมูล (Clustering) ที่ใช้ในการแบ่งข้อมูลที่มีคุณสมบัติคล้ายกันเป็นกลุ่ม (cluster) โดยใช้ค่าเฉลี่ยของข้อมูลในแต่ละกลุ่มเป็นจุดศูนย์กลาง (centroid) ของกลุ่มนั้น ๆ และจัดกลุ่มตามความใกล้เคียงของข้อมูลกับจุดศูนย์กลางนี้ การทำ K-Means สามารถสรุปได้ดังนี้:

- เลือกจำนวนกลุ่ม (K): ก่อนที่เราจะทำ K-Means, เราต้องกำหนดจำนวนของกลุ่มที่คุณต้องการให้แบ่งข้อมูลและมันคือพารามิเตอร์ K ใน K-Means. ทั้งนี้ในงานเราของเราได้วิธีการตัดสินใจด้วย Elbow Method โดยที่เมื่อค่า SSE ลดลงอย่างรวดเร็ว และหลังจากนั้นจึงค่อยๆ ลดลงนั้นคือจุดของจำนวนกลุ่มที่ต้องการ
- กำหนดจุดศูนย์กลางเริ่มต้น (Centroids): เราจะเลือกจุดศูนย์กลางเริ่มต้นสำหรับแต่ละกลุ่มอย่างสุ่มหรือจากการคำนวณตามข้อมูลตั้งต้น.
- กำหนดข้อมูลในกลุ่ม: ในขั้นตอนนี้, ข้อมูลจะถูกแบ่งออกเป็นกลุ่มตามความใกล้เคียงของข้อมูลกับจุดศูนย์กลาง. ข้อมูลจะถูกกำหนดให้อยู่ในกลุ่มที่มีจุดศูนย์กลางที่ใกล้ที่สุด.
- คำนวณจุดศูนย์กลางใหม่: จุดศูนย์กลางใหม่สำหรับแต่ละกลุ่มจะถูกคำนวณโดยหาค่าเฉลี่ยของข้อมูลในกลุ่มนั้น.
- ทำซ้ำขั้นตอน 3 และ 4: ขั้นตอนที่ 3 และ 4 จะถูกทำซ้ำจนกระทั่งจุดศูนย์กลางไม่เปลี่ยนแปลงมากและกลุ่มไม่มีการเปลี่ยนแปลงมาก.
- เราจะได้รับกลุ่มของข้อมูลที่ถูกแบ่งออกโดย K-Means และจุดศูนย์กลางสำหรับแต่ละกลุ่ม.

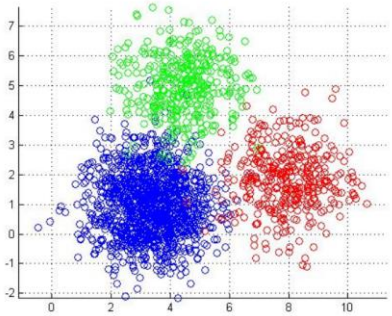

Figure 2 ตัวอย่าง K-means

3.1.2 Hierarchical Clustering เป็นเทคนิคในการจัดกลุ่มข้อมูลที่ทำได้ในรูปแบบลำดับหรือวงเวียน โดยอาศัยหลักการของความคล้ายคลึงของข้อมูลเพื่อสร้างโครงสร้างของกลุ่มแบบต้นไม้หรือ dendrogram ซึ่งแสดงความสัมพันธ์ระหว่างกลุ่มและสามารถที่จะแบ่งกลุ่มข้อมูลได้ในระดับที่ต่างกัน ขั้นตอนหลักในการทำ Hierarchical Clustering ได้แก่

- การคำนวณความคล้ายคลึง: ความคล้ายคลึงระหว่างข้อมูลสองตัวจะต้องถูกคำนวณ ซึ่งสามารถใช้วิธีการคำนวณระยะทาง (distance)
- สร้างกลุ่มเริ่มต้น: ในขั้นตอนแรก, แต่ละข้อมูลถือเป็นกลุ่มของตัวเอง
- การรวมกลุ่ม: กลุ่มที่มีข้อมูลที่คล้ายกันที่สุดจะถูกรวมเข้าด้วยกัน เรียกว่า "การรวม" (agglomeration) ในกรณีข้อมูลที่มีความคล้ายคลึงมากกัน
- ในขั้นตอนทุกครั้งที่มีการรวมกลุ่มใหม่จะมีการอัพเดต dendrogram ที่แสดงความสัมพันธ์ระหว่างกลุ่ม
- เมื่อทำซ้ำขั้นตอน 3 และ 4 จนกลุ่มเหลือแค่หนึ่งกลุ่มเดียว, เราจะได้รับ dendrogram สมบูรณ์ที่แสดงการจัดกลุ่มแบบลำดับของข้อมูล
- เราสามารถเลือกจำนวนของกลุ่มที่ต้องการจาก dendrogram โดยการตัดและส่งผ่านค่าความสูงบนดิสเจนดิแอกร์ (threshold) ที่ต้องการ

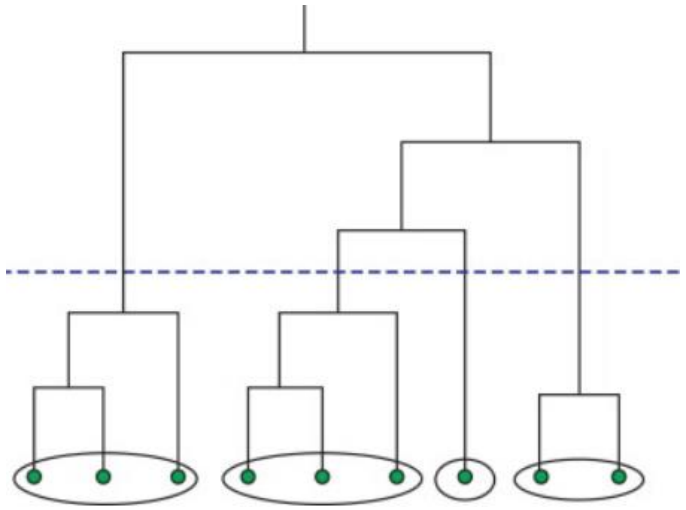

Figure 3 ตัวอย่าง Hierarchical Clustering.

3.1.3 DBScan คือวิธีการที่ใช้ความหนาแน่นของข้อมูลในพื้นที่เพื่อจัดกลุ่มข้อมูลที่อยู่ใกล้กันเข้าด้วยกัน และพิจารณาข้อมูลที่ห่างออกไปอย่างเด็ดขาดว่าเป็นข้อมูลเสีย (noise) โดยมีขั้นตอนดังนี้

- เราเลือกจุดข้อมูลใด ๆ ในชุดข้อมูลเป็นจุดเริ่มต้นด้วยวิธีการ Random
- คำนวณข้อมูลที่อยู่ใกล้จุดเริ่มต้นโดยใช้รัศมี (radius) ที่กำหนดไว้
- ถ้าจำนวนข้อมูลในรัศมีนี้มากกว่าหรือเท่ากับค่าที่กำหนด (MinPts) แล้วจุดเริ่มต้นถือเป็นจุดหนาแน่น (core point) และรัศมีนี้ถือเป็นพื้นที่หนาแน่น (dense region).
- การทำ DBSCAN จะเริ่มจากจุดเริ่มต้นที่เป็นจุดหนาแน่น และตรวจสอบข้อมูลที่ใกล้จุดเริ่มต้น และค้นหาข้อมูลที่อยู่ใกล้ๆกันและเป็นจุดหนาแน่นด้วย แล้วจะรวมกลุ่มข้อมูลเหล่านี้ในกลุ่มเดียวกัน. ขั้นตอนนี้ทำซ้ำจนกว่าจะไม่มีข้อมูลที่ถูกรวมกลุ่มเพิ่มเติม.
- เมื่อไม่มีข้อมูลที่รวมกลุ่มได้แล้วจะพบว่าข้อมูลที่เหลือเป็นข้อมูลเสีย (noise)

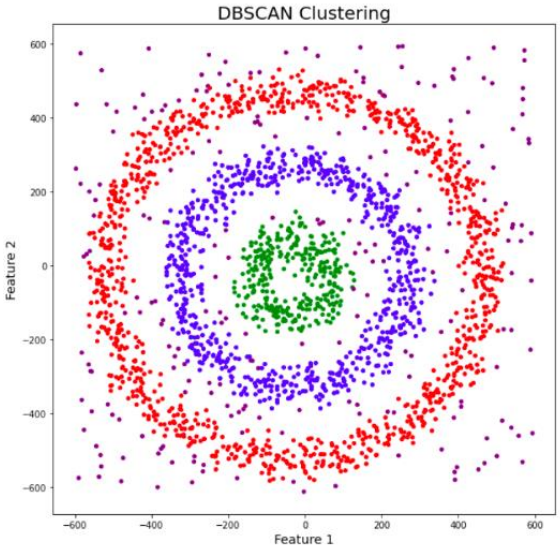

Figure 4 ตัวอย่าง DBScan

4. Methodology

ในการทำข้อมูลที่เปรียบเทียบในครั้งนี้ได้ใช้ชุดข้อมูลสาธารณะจาก Kaggle ภายใต้ชุดข้อมูล Credit Card Customer Data สำหรับการทำ Clustering กรณีที่ไม่ได้ทำ PCA และกรณีที่ทำ PCA โดยมีขั้นตอนดังนี้

1. Data Preparation
2. PCA
3. K-means ด้วย Euclidean Distance ทั้งกรณีที่ไม่ได้ทำ PCA และกรณีที่ทำ PCA
4. K-means ด้วย Manhattan Distance ทั้งกรณีที่ไม่ได้ทำ PCA และกรณีที่ทำ PCA
5. Hierarchical Clustering ด้วย Euclidean Distance ทั้งกรณีที่ไม่ได้ทำ PCA และกรณีที่ทำ PCA
6. Hierarchical Clustering ด้วย Manhattan Distance ทั้งกรณีที่ไม่ได้ทำ PCA และกรณีที่ทำ PCA
7. DBScan ด้วย Euclidean Distance ทั้งกรณีที่ไม่ได้ทำ PCA และกรณีที่ทำ PCA
8. DBScan ด้วย Manhattan Distance ทั้งกรณีที่ไม่ได้ทำ PCA และกรณีที่ทำ PCA

เมื่อทำทุกขั้นตอนเสร็จจะทำการเปรียบเทียบกรณีที่ไม่ได้ทำ PCA และกรณีที่ทำ PCA ในแต่ละวิธี

4.1 Data Preparation

ขั้นตอนที่ 1 : ทำการเรียกใช้ Package ทั้งหมด 8 ตัวได้แก่

```
library(DataExplorer)
library(factoextra)
library(corrplot)
library(ClusterR)
library(cluster)
library(ggfortify)
library(stats)
library(fpc)
```

Figure 5 package ทั้งหมดสำหรับการใช้งาน

DataExplorer: แพ็คเกจนี้มีเครื่องมือสำหรับวิเคราะห์ข้อมูล ที่ช่วยในการสร้างกราฟและสรุปข้อมูลในชุดข้อมูล มีฟังก์ชันที่ช่วยในการสร้างกราฟและรายงานเชิงสถิติเบื้องต้น

factoextra: แพ็คเกจนี้เป็นเครื่องมือสำหรับการวิเคราะห์ข้อมูลหลายตัวแปร (multivariate data analysis) ซึ่งช่วยในการสร้างกราฟและสรุปผลลัพธ์จากการวิเคราะห์แฟคทอริเชียล (factorial analysis)

corrplot: แพ็คเกจนี้ช่วยในการสร้างกราฟแสดงความสัมพันธ์ระหว่างตัวแปรในชุดข้อมูล โดยแสดงผลเป็นเมทริกซ์แสดงความสัมพันธ์

ClusterR: แพ็คเกจนี้เกี่ยวกับการแบ่งกลุ่ม (clustering) ซึ่งช่วยในการหากลุ่มที่คล้ายกันในชุดข้อมูล

cluster: แพ็คเกจนี้ใช้สำหรับการวิเคราะห์แบบจำนวนไม่มากของกลุ่มในชุดข้อมูล โดยใช้เทคนิคต่าง ๆ เช่น hierarchical clustering หรือ k-means clustering

ggfortify: แพ็คเกจนี้ช่วยในการนำข้อมูลที่ถูกวิเคราะห์ด้วยโมเดลสถิติมาแสดงผลในรูปแบบกราฟโดยใช้ ggplot2

stats: แพ็คเกจที่มาพร้อมกับ R โดยปริยายเพิ่มเติมสถิติพื้นฐานและฟังก์ชันสถิติต่าง ๆ เพื่อใช้ในการวิเคราะห์ข้อมูล

fpc: แพ็คเกจนี้ใช้ในการวิเคราะห์ข้อมูลเชิงกลุ่ม (cluster analysis) โดยให้ความสำคัญกับการหาจำนวนกลุ่มที่เหมาะสมในการแบ่งกลุ่มข้อมูล

ขั้นตอนที่ 2 : นำข้อมูลเข้ามายัง R ด้วยวิธีการกำหนดเส้นทางในคอมพิวเตอร์เราจากนั้นใช้คำสั่ง read.csv(file) ในการอ่านข้อมูลที่นำเข้า

```
file = 'C:/Users/User/Documents/data/Credit Card Customer Data.csv'
data = read.csv(file)
data
```

Sl_No	Customer.Key	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made
1	87073	100000	2	1	1	0
2	38414	50000	3	0	10	9
3	17341	50000	7	1	3	4
4	40496	30000	5	1	1	4
5	47437	100000	6	0	12	3
6	58634	20000	3	0	1	8
7	48370	100000	5	0	11	2
8	37376	15000	3	0	1	1
9	82490	5000	2	0	2	2
10	44770	3000	4	0	1	7
11	52741	10000	4	0	5	5
12	52326	13000	3	0	2	7
13	92503	11000	1	2	5	9
14	25084	9000	1	1	5	6
15	68517	6000	2	2	4	6
16	55196	8000	2	0	5	7
17	62617	15000	2	1	2	4

Figure 6 การนำข้อมูลเข้าและตัวอย่างข้อมูล

ขั้นตอนที่ 3 : ทำการตรวจสอบ Missing Value จำนวน rows และ colums ได้จาก Introduce(data)

พบว่ามีข้อมูลทั้งหมด 4620 ข้อมูลโดยที่มี 660 rows และ 7 columns และค่า Missing Value เป็น 0 เราจึงสามารถนำข้อมูลมาใช้ได้

```
introduce(data)
```

```
  rows columns discrete_columns continuous_columns all_missing_columns total_missing_values
1  660       7                0                  7                   0                    0
  complete_rows total_observations memory_usage
1           660               4620        21016
```

Figure 7 การใช้คำสั่งตรวจสอบข้อมูลและผลลัพธ์

ขั้นตอนที่ 4 : ทำการเลือกข้อมูลที่ใช้สำหรับการวิเคราะห์ columns 2 ถึง 7 ด้วย data[,2:7]

```
new_df <- data[,2:7]
new_df
```

Customer.Key	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made
87073	100000	2	1	1	0
38414	50000	3	0	10	9
17341	50000	7	1	3	4
40496	30000	5	1	1	4
47437	100000	6	0	12	3
58634	20000	3	0	1	8
48370	100000	5	0	11	2
37376	15000	3	0	1	1
82490	5000	2	0	2	2
44770	3000	4	0	1	7
52741	10000	4	0	5	5
52326	13000	3	0	2	7
92503	11000	1	2	5	9
25084	9000	1	1	5	6
68517	6000	2	2	4	6
55196	8000	2	0	5	7
62617	15000	2	1	2	4

Figure 8 การเลือกตัวแปรที่ใช้งาน

ขั้นตอนที่ 5 : ปรับข้อมูลให้เป็น Standardize ด้วย Scale(new_df)

```
scale_data <- scale(new_df)
scale_data
```

Customer.Key	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made
1.245974711	1.7388680	-1.2482780	-0.8597985	-0.5470748	-1.2505889
-0.652707688	0.4099816	-0.7869883	-1.4726139	2.5186084	1.8904250
-1.474979707	0.4099816	1.0581707	-0.8597985	0.1341882	0.1454173
-0.571467696	-0.1215730	0.1355912	-0.8597985	-0.5470748	0.1454173
-0.300628704	1.7388680	0.5968810	-1.4726139	3.1998713	-0.2035842
0.136280127	-0.3873503	-0.7869883	-1.4726139	-0.5470748	1.5414235
-0.264222886	1.7388680	0.1355912	-1.4726139	2.8592399	-0.5525858
-0.693210623	-0.5202389	-0.7869883	-1.4726139	-0.5470748	-0.9015873
1.067145277	-0.7860162	-1.2482780	-1.4726139	-0.2064433	-0.5525858
-0.404695494	-0.8391716	-0.3256985	-1.4726139	-0.5470748	1.1924219
-0.093665728	-0.6531275	-0.3256985	-1.4726139	0.8154511	0.4944189
-0.109859098	-0.5733944	-0.7869883	-1.4726139	-0.2064433	1.1924219
1.457854228	-0.6265498	-1.7095678	-0.2469832	0.8154511	1.8904250
-1.172846539	-0.6797053	-1.7095678	-0.8597985	0.8154511	0.8434204
0.521916457	-0.7594385	-1.2482780	-0.2469832	0.4748196	0.8434204
0.002128787	-0.7062830	-1.2482780	-1.4726139	0.8154511	1.1924219
0.291697460	-0.5202389	-1.2482780	-0.8597985	-0.2064433	0.1454173

Figure 9 ตัวอย่างข้อมูลที่ปรับสเกลแล้ว

4.2 PCA

ขั้นตอนที่ 1 : ทำการปรับข้อมูลให้อยู่ในรูปแบบ PCA ด้วย prcomp(scale_data, scale = TRUE)

พบว่าข้อมูลมีดังนี้ ภายใต้เงื่อนไขเกณฑ์การตัดสินใจเมื่อ Cumulative ที่มากกว่า 80% ยอมรับข้อมูลสูญหายไป 20% อยู่ที่ PC3 หมายความว่าจะคัดเลือก PC ตั้งแต่ 1 – 3

```
res.pca = prcomp(scale_data, scale = TRUE)
print(summary(res.pca))
```

```
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6
Standard deviation     1.5124 1.3688 1.0005 0.56568 0.52565 0.49181
Proportion of Variance 0.3812 0.3123 0.1668 0.05333 0.04605 0.04031
Cumulative Proportion  0.3812 0.6935 0.8603 0.91364 0.95969 1.00000
```

Figure 10 ผลลัพธ์จากการคำนวณ PCA

จากนั้นทำการ plot ค่า Eigenvalue เพื่อดูน้ำหนักของแต่ละ PC ด้วยคำสั่ง fviz_eig(res.pca,addlabels = TRUE)

```
fviz_eig(res.pca,addlabels = TRUE)
```

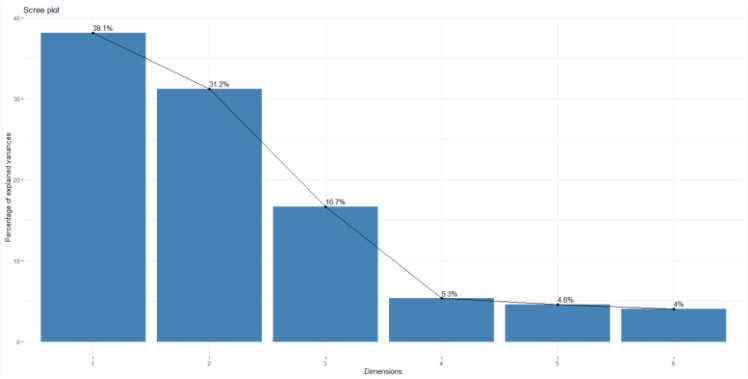

Figure 11 กราฟแสดง Eigenvalue ของแต่ละ PC

ขั้นตอนที่ 2 : ทำ Correlation Circle ด้วยการเก็บตัวแปรค่า pca ไว้ที่ var ด้วยคำสั่ง get_pca_var(res.pca)

```
var = get_pca_var(res.pca)
```

Figure 12 เก็บค่า PCA

ขั้นตอนที่ 3 : ทำการ plot ค่าความสัมพันธ์ของแต่ละทิศทางด้วย fviz_pca_var(res.pca, col.var =”black”)

```
fviz_pca_var(res.pca, col.var = "black")
```

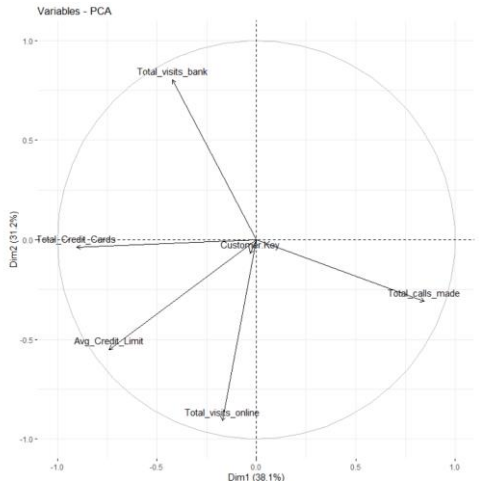

Figure 13 กราฟ Correlation circle สำหรับดูทิศทาง

พบว่าเราสามารถอธิบายข้อมูลที่สำคัญโดยยกตัวอย่าง 3 ตัวแปรได้ว่า

- ค่าเฉลี่ยเงินในบัตรเครดิต (Avg_credit_Limit) น้อยก็จะมีการใช้เงินที่น้อยตามไปด้วย
- จำนวนครั้งที่เข้าใช้เงินทางออนไลน์ (Total_visit_online) น้อยแสดงว่าจะมีการใช้เงินน้อยไปด้วย
- จำนวนครั้งที่เข้าใช้เงินทางธนาคาร (Total_visit_online) มากแสดงว่าจะมีการใช้เงินน้อยไปด้วย

ขั้นตอนที่ 4 : ทำการ plot ค่าความสัมพันธ์ด้วย corrplot(var$cos2, is.corr = FALSE)

```
corrplot(var$cos2, is.corr = FALSE)
```

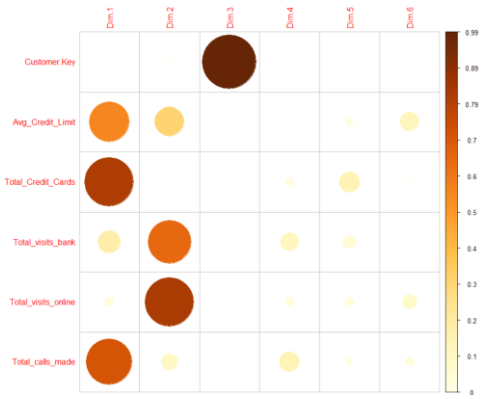

Figure 14 กราฟแสดงระดับความสัมพันธ์

พบว่าได้ความสัมพันธ์ดังตารางนี้ทั้งนี้จะอธิบาย Dimension 1 ดังนี้

- ความสัมพันธ์ระหว่าง Dimension 1 กับ Customer.Key ไม่มีความสัมพันธ์กันเลย
- ความสัมพันธ์ระหว่าง Dimension 1 กับ Avg_Credit_limit มีความสัมพันธ์กันปานกลาง
- ความสัมพันธ์ระหว่าง Dimension 1 กับ Total_Credit_Card มีความสัมพันธ์กันมาก
- ความสัมพันธ์ระหว่าง Dimension 1 กับ Total_Visit_bank มีความสัมพันธ์กันน้อย
- ความสัมพันธ์ระหว่าง Dimension 1 กับ Total_Visit_online มีความสัมพันธ์กันน้อย
- ความสัมพันธ์ระหว่าง Dimension 1 กับ Total_calls_made มีความสัมพันธ์กันมาก

ขั้นตอนที่ 4 : ทำการ ค่าความสัมพันธ์ของแต่ละทิศทางและมีความสัมพันธ์มากน้อยขนาดไหนด้วย fviz_pca_var(res.pca, col.var = "cos2",gradient.cols = c("#00AFBB","#E7B880","#FC4E07"))

```
fviz_pca_var(res.pca, col.var = "cos2",
             gradient.cols = c("#00AFBB","#E7B880","#FC4E07"))
```

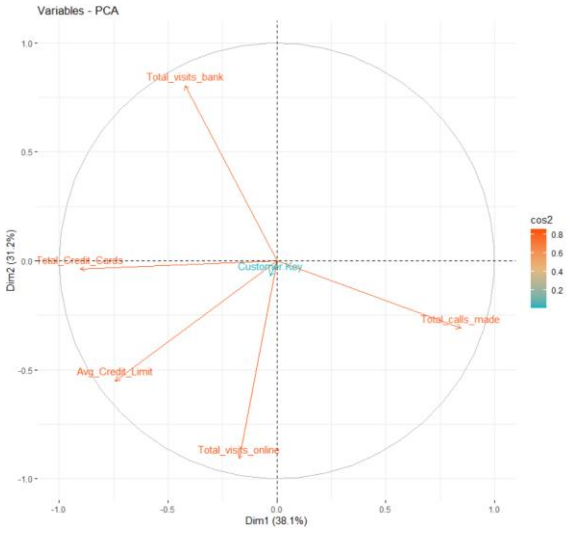

Figure 15 กราฟแสดงความทิศทางและระดับของความสัมพันธ์

พบว่าเราสามารถอธิบายข้อมูลที่สำคัญโดยยกตัวอย่าง 3 ตัวแปรได้ว่า

- ค่าเฉลี่ยเงินในบัตรเครดิต (Avg_credit_Limit) น้อยก็จะมีการใช้เงินที่น้อยตามไปด้วยโดยมีความสัมพันธ์อยู่ในระดับมาก

- จำนวนครั้งที่เข้าใช้เงินทางออนไลน์ (Total_visit_online) น้อยแสดงว่าจะมีการใช้เงินน้อยไปด้วยโดยมีความสัมพันธ์อยู่ในระดับมาก
- จำนวนครั้งที่เข้าใช้เงินทางธนาคาร (Total_visit_online) มากแสดงว่าจะมีการใช้เงินน้อยไปด้วยโดยมีความสัมพันธ์อยู่ในระดับมาก

ขั้นตอนที่ 5 : เราสามารถใช้คำสั่ง ind <- get_pca_ind(res.pca) fviz_pca_ind(res.pca, col.ind = "cos2", gradient.cols = c("#00AFBB","#E7B880","#FC4E07")) เพื่อทำการหาว่าแต่ละคนมีความสัมพันธ์กันมากน้อยขนาดไหน

```
ind <- get_pca_ind(res.pca)
fviz_pca_ind(res.pca, col.ind = "cos2",
             gradient.cols = c("#00AFBB","#E7B880","#FC4E07"))
```

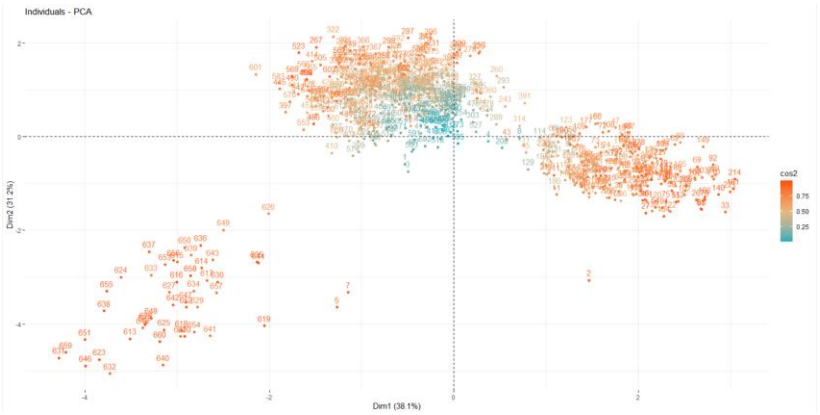

Figure 16 กราฟสำหรับตรวจสอบความสัมพันธ์ของแต่ละ datapoint

ขั้นตอนที่ 6 : กำหนดตัวแปรไว้ทั้งหมด 2 ตัวสำหรับการใช้งานประกอบด้วย

- scale_data สำหรับกรณีไม่ได้ทำ PCA
- pca_data สำหรับกรณีทำ PCA

```
pca_data <- res.pca$x[, 1:3]
scale_data # กรณีไม่ได้ทำ PCA
pca_data # กรณีทำ PCA
```

Figure 17 กำหนดตัวแปรสำหรับการใช้งาน

4.3 K-means ด้วย Euclidean Distance

กรณีที่ไม่ได้ทำ PCA

ขั้นตอนที่ 1 : หา K จากทำการหาค่า Elbow method ด้วยวิธีการ loop หาค่า SSE โดยกำหนดกลุ่มตั้งแต่ 1-20 ด้วยวิธีการหาระยะทางแบบ Euclidean distance และใช้ข้อมูลของกรณีไม่ได้ทำ PCA จากนั้นทำการ plot ด้วย ggplot โดยมีการตัดสินใจเมื่อ SSE ลดลงอย่างลดเร็วและค่อยๆลดลงในระยะหลังจึงจะเลือก K เท่ากับกลุ่มนั้น จากกราฟพบว่ากลุ่มที่ครบตามเงื่อนไขคือ 3 จึงให้ K มีค่าเท่ากับ 3

```
sse <-numeric(20)
for (k in 1:20) {
  kmeans_model <- kmeans(scale_data, centers = k)
  sse[k] <- kmeans_model$tot.withinss
}

ggplot(data.frame(K = 1:20, SSE = sse), aes(x = K, y = SSE)) +
  geom_line() +
  geom_point() +
  labs(x = "จำนวนกลุ่ม (K)", y = "SSE") +
  theme_minimal()
```

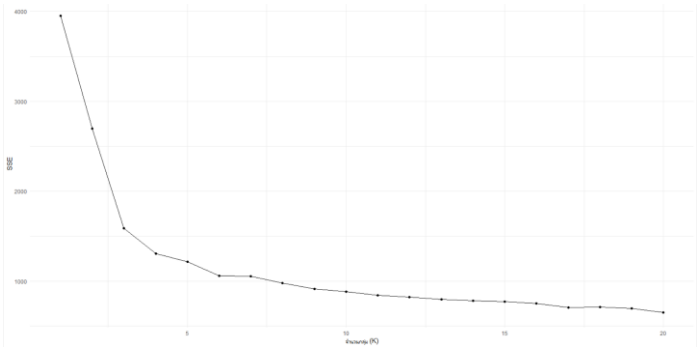

Figure 18 การคัดเลือก K

ขั้นตอนที่ 2 : จัดหากลุ่มวิธีการ kmeans ด้วยคำสั่ง kmeans(scale_data,center = 3)

พบว่าจะการทำวิธีการ K-means สามารถจัด 3 กลุ่มได้แก่ 386,224 และ 50 ทั้งนี้พบว่าค่า total sum of square อยู่ที่ 59.8%

```
kmeans_result <- kmeans(scale_data,center = 3)
kmeans_result
```

```
K-means clustering with 3 clusters of sizes 386, 50, 224

Within cluster sum of squares by cluster:
[1] 930.9044 168.1350 491.3606
 (between_SS / total_SS =  59.8 %)
```

Figure 19 แสดงค่าผลลัพธ์ของ K-means

ขั้นตอนที่ 3 : ทำการ plot เพื่อดู visualization ของข้อมูลด้วย autoplot(kmeans_result,scale_data,frame=TRUE)

```
autoplot(kmeans_result,scale_data,frame=TRUE)
```

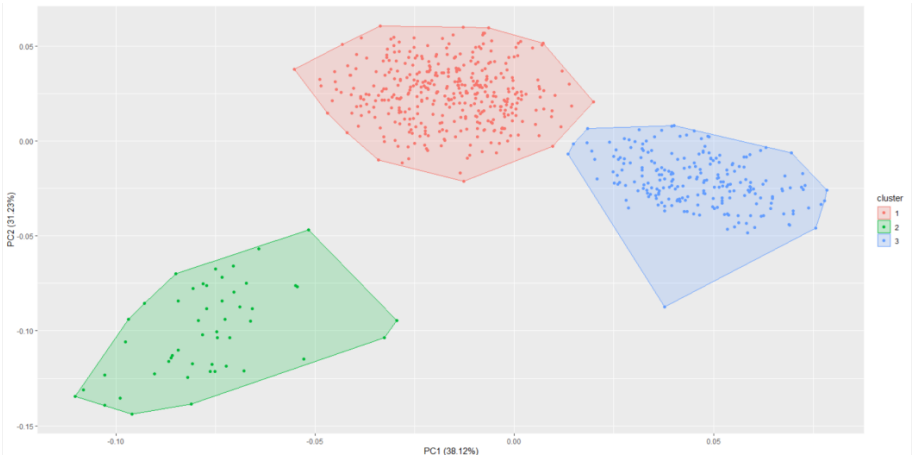

Figure 20 การ plot ของ K-means

สรุปผลจากขั้นตอนทั้งหมดสามารถสรุปได้ว่าการทำ K-means ด้วยวิธีการ Euclidean distance จากข้อมูลกรณีไม่ได้ทำ PCA พบว่าเราได้จัดกลุ่มทั้งหมด 3 กลุ่มด้วยวิธีการ Elbow method ได้แก่ 386,224 และ 50 ทั้งนี้พบว่าค่า total sum of square อยู่ที่ 59.8%

กรณีที่ทำ PCA

ขั้นตอนที่ 1 : หา K จากทำการหาค่า Elbow method ด้วยวิธีการ loop หาค่า SSE โดยกำหนดกลุ่มตั้งแต่ 1-20 ด้วยวิธีการหาระยะทางแบบ Euclidean distance และใช้ข้อมูลของกรณีทำ PCA จากนั้นทำการ plot ด้วย ggplot โดยมีการตัดสินใจเมื่อ SSE ลดลงอย่างลดเร็วและค่อยๆลดลงในระยะหลังจึงจะเลือก K เท่ากับกลุ่มนั้น

จากกราฟพบว่ากลุ่มที่ครบตามเงื่อนไขคือ 3 จึงให้ K มีค่าเท่ากับ 3

```
sse <-numeric(20)
for (k in 1:20) {
  kmeans_model <- kmeans(pca_data, centers = k)
  sse[k] <- kmeans_model$tot.withinss
}

ggplot(data.frame(K = 1:20, SSE = sse), aes(x = K, y = SSE)) +
  geom_line() +
  geom_point() +
  labs(x = "จำนวนกลุ่ม (K)", y = "SSE") +
  theme_minimal()
```

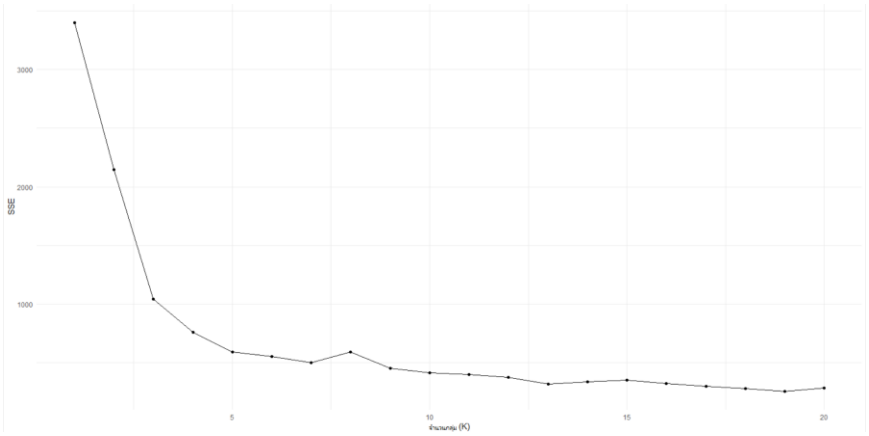

Figure 21 การคัดเลือก K

ขั้นตอนที่ 2 : จัดหากลุ่มวิธีการ kmeans ด้วยคำสั่ง kmeans(pca_data,center = 3)

พบว่าจะการทำวิธีการ K-means สามารถจัด 3 กลุ่มได้แก่ 386,224 และ 50 ทั้งนี้พบว่าค่า total sum of square อยู่ที่ 69.4%

```
kmeans_result_p <- kmeans(pca_data,center = 3)
kmeans_result_p
```

```
K-means clustering with 3 clusters of sizes 386, 224, 50

Within cluster sum of squares by cluster:
[1] 594.1824 340.8401 106.9067
 (between_SS / total_SS =  69.4 %)
```

Figure 22 แสดงค่าผลลัพธ์ของ K-means

ขั้นตอนที่ 3 : ทำการ plot เพื่อดู visualization ของข้อมูลด้วย autoplot(kmeans_result_p,pca_data,frame=TRUE)

```
autoplot(kmeans_result_p,pca_data,frame=TRUE)
```

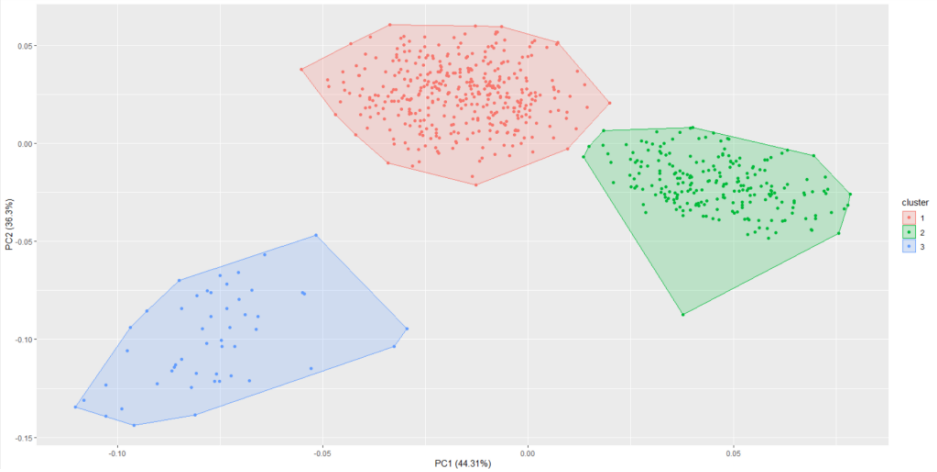

Figure 23 การ plot ของ K-means

สรุปผลจากขั้นตอนทั้งหมดสามารถสรุปได้ว่าการทำ K-means ด้วยวิธีการ Euclidean distance จากข้อมูลกรณีทำ PCA พบว่าเราได้จัดกลุ่มทั้งหมด 3 กลุ่มด้วยวิธีการ Elbow method ได้แก่ 386,224 และ 50 ทั้งนี้พบว่าค่า total sum of square อยู่ที่ 69.4%

4.4 K-means ด้วย Manhattan Distance

กรณีที่ไม่ได้ทำ PCA

ขั้นตอนที่ 1 : หา K จากทำการหาค่า Elbow method ด้วยวิธีการ loop หาค่า SSE โดยกำหนดกลุ่มตั้งแต่ 1-20 ด้วยวิธีการหาระยะทางแบบ Manhattan distance จากการกำหนด algorithm เป็น Lloyd และใช้ข้อมูลของกรณีไม่ได้ทำ PCA จากนั้นทำการ plot ด้วย ggplot โดยมีการตัดสินใจเมื่อ SSE ลดลงอย่างลดเร็วและค่อยๆ ลดลงในระยะหลังจึงจะเลือก K เท่ากับกลุ่มนั้น

จากกราฟพบว่ากลุ่มที่ครบตามเงื่อนไขคือ 3 จึงให้ K มีค่าเท่ากับ 3

```
sse <-numeric(20)
for (k in 1:20) {
  kmeans_model <- kmeans(scale_data, centers = k,algorithm = "Lloyd")
  sse[k] <- kmeans_model$tot.withinss
}

ggplot(data.frame(K = 1:20, SSE = sse), aes(x = K, y = SSE)) +
  geom_line() +
  geom_point() +
  labs(x = "จำนวนกลุ่ม (K)", y = "SSE") +
  theme_minimal()
```

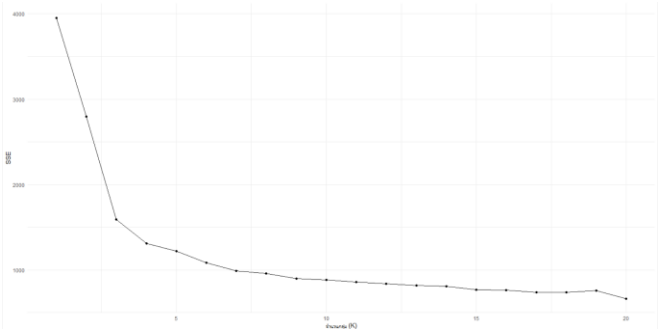

Figure 24 การคัดเลือก K

ขั้นตอนที่ 2 : จัดหากลุ่มวิธีการ kmeans ด้วยคำสั่ง kmeans(scale_data,center = 3,algorithm = “Lloyd”)

พบว่าจะการทำวิธีการ K-means สามารถจัด 3 กลุ่มได้แก่ 386,224 และ 50 ทั้งนี้พบว่าค่า total sum of square อยู่ที่ 59.8%

```
kmeans_result_m <- kmeans(scale_data,center = 3,algorithm = "Lloyd")
kmeans_result_m
K-means clustering with 3 clusters of sizes 386, 224, 50

Within cluster sum of squares by cluster:
[1] 930.9044 491.3606 168.1350
 (between_SS / total_SS =  59.8 %)
```

Figure 25 แสดงค่าผลลัพธ์ของ K-means

ขั้นตอนที่ 3 : ทำการ plot เพื่อดู visualization ของข้อมูลด้วย autoplot(kmeans_result_m,scale_data,frame=TRUE)

```
autoplot(kmeans_result_m,scale_data,frame=TRUE)
```

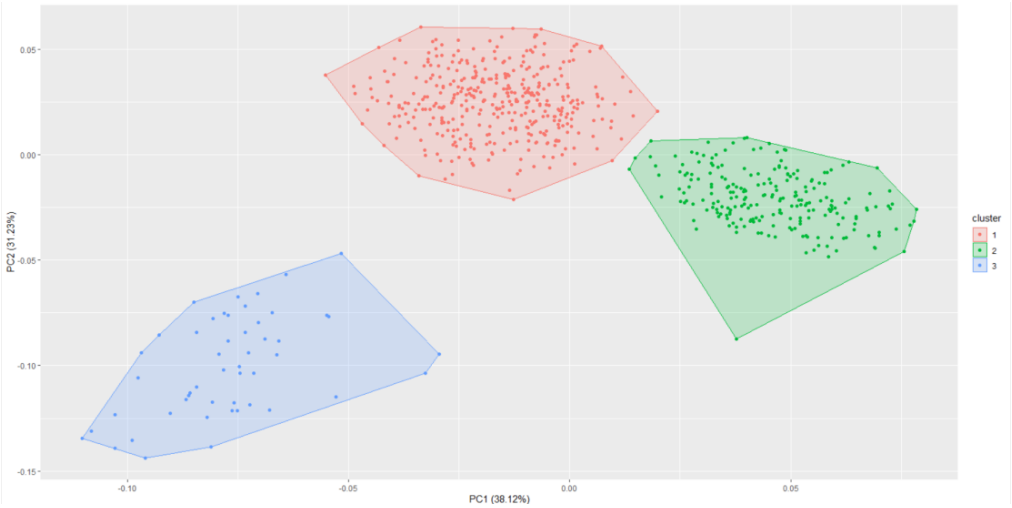

Figure 26 การ plot ของ K-means

สรุปผลจากขั้นตอนทั้งหมดสามารถสรุปได้ว่าการทำ K-means ด้วยวิธีการ Manhattan distance จากข้อมูลกรณีไม่ได้ทำ PCA พบว่าเราได้จัดกลุ่มทั้งหมด 3 กลุ่มด้วยวิธีการ Elbow method ได้แก่ 386,224 และ 50 ทั้งนี้พบว่าค่า total sum of square อยู่ที่ 59.8%

กรณีที่ทำ PCA

ขั้นตอนที่ 1 : หา K จากทำการหาค่า Elbow method ด้วยวิธีการ loop หาค่า SSE โดยกำหนดกลุ่มตั้งแต่ 1-20 ด้วยวิธีการหาระยะทางแบบ Manhattan distance จากการกำหนด algorithm เป็น Lloyd และใช้ข้อมูลของกรณีไม่ได้ทำ PCA จากนั้นทำการ plot ด้วย ggplot โดยมีการตัดสินใจเมื่อ SSE ลดลงอย่างลดเร็วและค่อยๆ ลดลงในระยะหลังจึงจะเลือก K เท่ากับกลุ่มนั้น

จากกราฟพบว่ากลุ่มที่ครบตามเงื่อนไขคือ 4 จึงให้ K มีค่าเท่ากับ 4

```
sse <-numeric(20)
for (k in 1:20) {
  kmeans_model <- kmeans(pca_data, centers = k,algorithm = "Lloyd")
  sse[k] <- kmeans_model$tot.withinss
}

ggplot(data.frame(K = 1:20, SSE = sse), aes(x = K, y = SSE)) +
  geom_line() +
  geom_point() +
  labs(x = "จำนวนกลุ่ม (K)", y = "SSE") +
  theme_minimal()
```

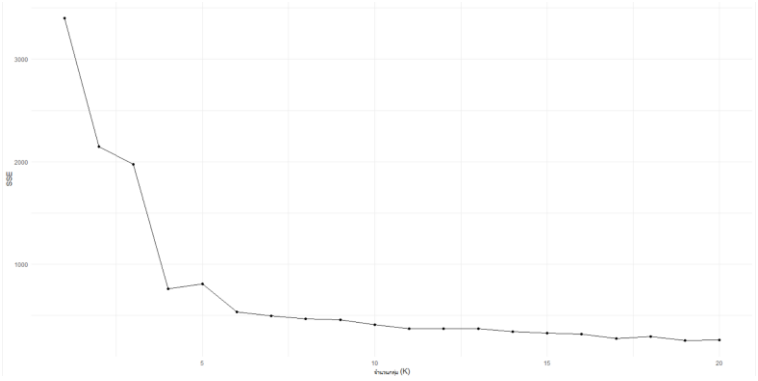

Figure 27 การคัดเลือก K

ขั้นตอนที่ 2 : จัดหากลุ่มวิธีการ kmeans ด้วยคำสั่ง kmeans(pca_data,center = 4,algorithm = “Lloyd”)

พบว่าจะการทำวิธีการ K-means สามารถจัด 4 กลุ่มได้แก่ 223,50,172 และ 215 ทั้งนี้พบว่าค่า total sum of square อยู่ที่ 77.7%

```
kmeans_result_pm <- kmeans(pca_data,center = 4,algorithm = "Lloyd")
kmeans_result_pm

K-means clustering with 4 clusters of sizes 223, 50, 172, 215

Within cluster sum of squares by cluster:
[1] 336.6870 106.9067 137.2065 178.5276
 (between_SS / total_SS =  77.7 %)
```

Figure 28 แสดงค่าผลลัพธ์ของ K-means

ขั้นตอนที่ 3 : ทำการ plot เพื่อดู visualization ของข้อมูลด้วย autoplot(kmeans_result_pm,pca_data,frame=TRUE)

```
autoplot(kmeans_result_pm,pca_data,frame=TRUE)
```

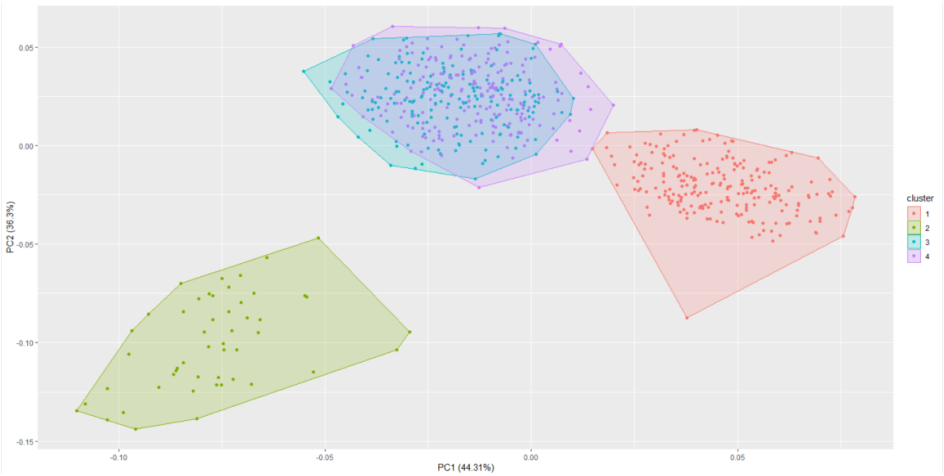

Figure 29 การ plot ของ K-means

สรุปผลจากขั้นตอนทั้งหมดสามารถสรุปได้ว่าการทำ K-means ด้วยวิธีการ Manhattan distance จากข้อมูล กรณีทำ PCA พบว่าเราได้จัดกลุ่มทั้งหมด 4 กลุ่มด้วยวิธีการ Elbow method ได้แก่ 223,50,172 และ 215 ทั้งนี้ พบว่าค่า total sum of square อยู่ที่ 77.7%

4.5 Hierarchical Clustering ด้วย Euclidean Distance

ก่อนการทำ Hierarchical Clustering จะกำหนดข้อมูลที่นำมาใช้ทั้งหมด 50 ข้อมูล

```
test = (scale_data[1:50,])
test

test_pca = (pca_data[1:50,])
test_pca
```

Figure 30 กำหนดข้อมูลสำหรับ Hierarchical Clustering

กรณีที่ไม่ได้ทำ PCA

ขั้นตอนที่ 1 : กำหนดระยะทางที่จะใช้ด้วยและข้อมูลที่ใช้ด้วย dist(test,method = 'euclidean') จากนั้นนำไปใช้ใน hclust เพื่อทำการสร้างกลุ่มแบบ Hierarchical Clustering

```
distance_mat <- dist(test,method = 'euclidean')
distance_mat
Hierar_cl <- hclust(distance_mat)
Hierar_cl
```

Figure 31 คำนวณระยะห่างและจัดกลุ่ม Heirarchical

ขั้นตอนที่ 2 : ทำเช็คข้อมูลจากการรันและทำการ plot ในรูปแบบ Dendrogram

```
Call:
hclust(d = distance_mat)

Cluster method   : complete
Distance         : euclidean
Number of objects: 50

plot(Hierar_cl)
```

Figure 32 ข้อมูลการรันและการสร้างกราฟ Hierarchical

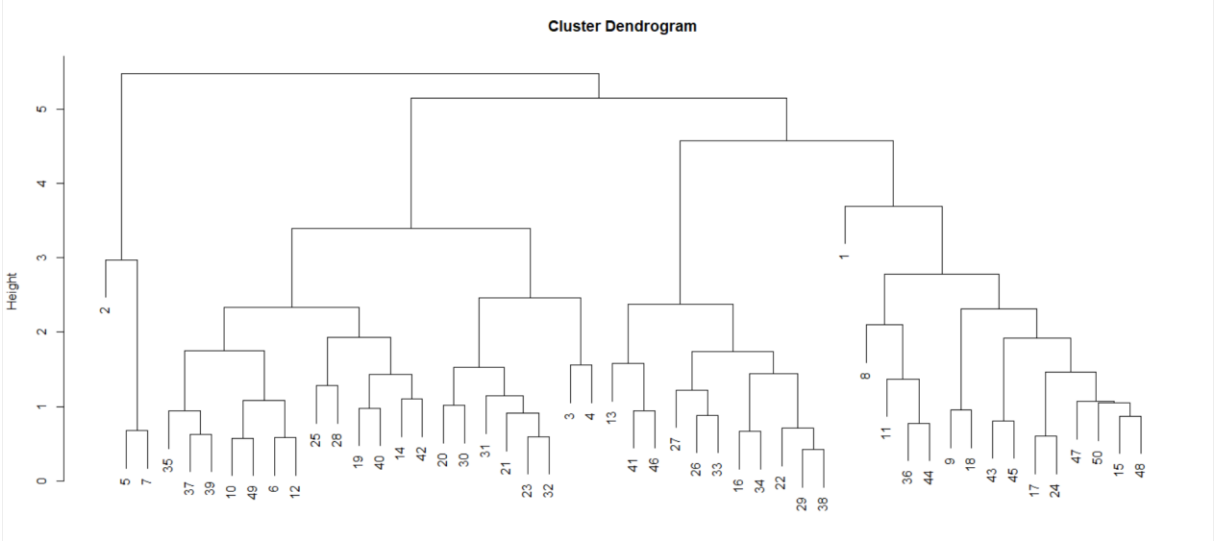

Figure 33 Dendrogram

ขั้นตอนที่ 3 : จากผลลัพธ์ที่ได้ใน dendrogram จึงจัดสินใจด้วยการเลือกจำนวนกลุ่มทั้งหมด 3 กลุ่ม โดยตัดกลุ่มที่ระดับความสูง 4.75

```
plot(Hierar_cl)
abline(h=4.75,col = "green")
```

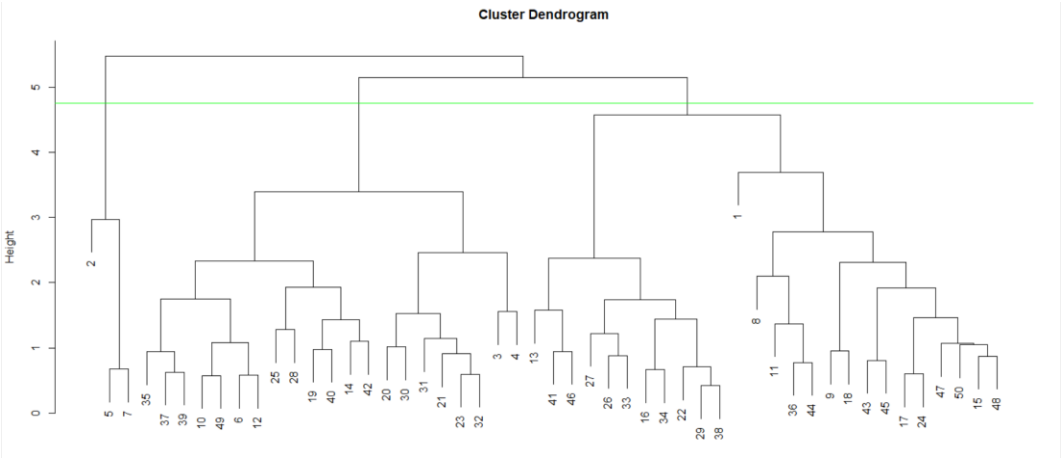

Figure 34 จุดตัดสินใจสำหรับการสร้างกลุ่ม

ขั้นตอนที่ 4 : เราจึงทำการแบ่งกลุ่มเท่ากับ 3 กลุ่มโดยที่ใช้ fit = 3 เพื่อแสดงผลของแต่ละกลุ่มจากนั้น plot ด้วยคำสั่ง plot

พบแบ่งกลุ่มได้ 3 กลุ่มได้แก่ 26,3 และ 21

```
fit<- cutree(Hierar_cl,k=3)
fit
plot(Hierar_cl)
table(fit)
rect.hclust(Hierar_cl,k=3,border = "green")
```

```
fit
 1  2  3
26  3 21
```

Figure 35 โค้ดและจำนวนของการจัดกลุ่ม

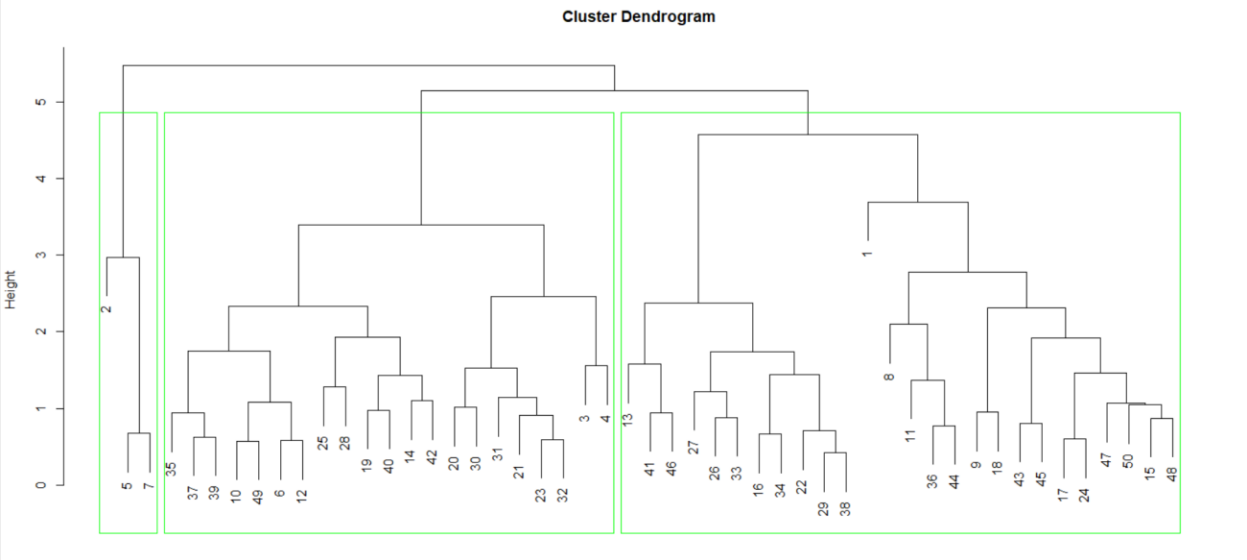

Figure 36 Dendrogram ที่จัดกลุ่มเรียบร้อยแล้ว

สรุปผลจากขั้นตอนทั้งหมดสามารถสรุปได้ว่าการทำ Hierarchical Clustering ด้วยวิธีการ Euclidean distance จากข้อมูลกรณีไม่ได้ทำ PCA จำนวน 50 ข้อมูลพบว่าภายใต้การตัดสินใจจากการทำ Dendrogram จึงตัดสินใจจัดกลุ่ม 3 ที่ความสูง 4.75 ประกอบไปด้วยได้แก่ 26,3 และ 21

กรณีที่ทำ PCA

ขั้นตอนที่ 1 : กำหนดระยะทางที่จะใช้ด้วยและข้อมูลที่ใช้ด้วย dist(test_pca,method = 'euclidean') จากนั้นนำไปใช้ใน hclust เพื่อทำการสร้างกลุ่มแบบ Hierarchical Clustering

```
distance_mat_p <- dist(test_pca,method = 'euclidean')
distance_mat_p
Hierar_cl_p <- hclust(distance_mat_p)
Hierar_cl_p
```

Figure 37 คำนวณระยะห่างและจัดกลุ่ม Heirarchical

ขั้นตอนที่ 2 : ทำเช็คข้อมูลจากการรันและทำการ plot ในรูปแบบ Dendrogram

```
Call:
hclust(d = distance_mat_p)

Cluster method   : complete
Distance         : euclidean
Number of objects: 50
```

```
plot(Hierar_cl_p)
```

Figure 38 ข้อมูลการรันและการสร้างกราฟ Hierarchical

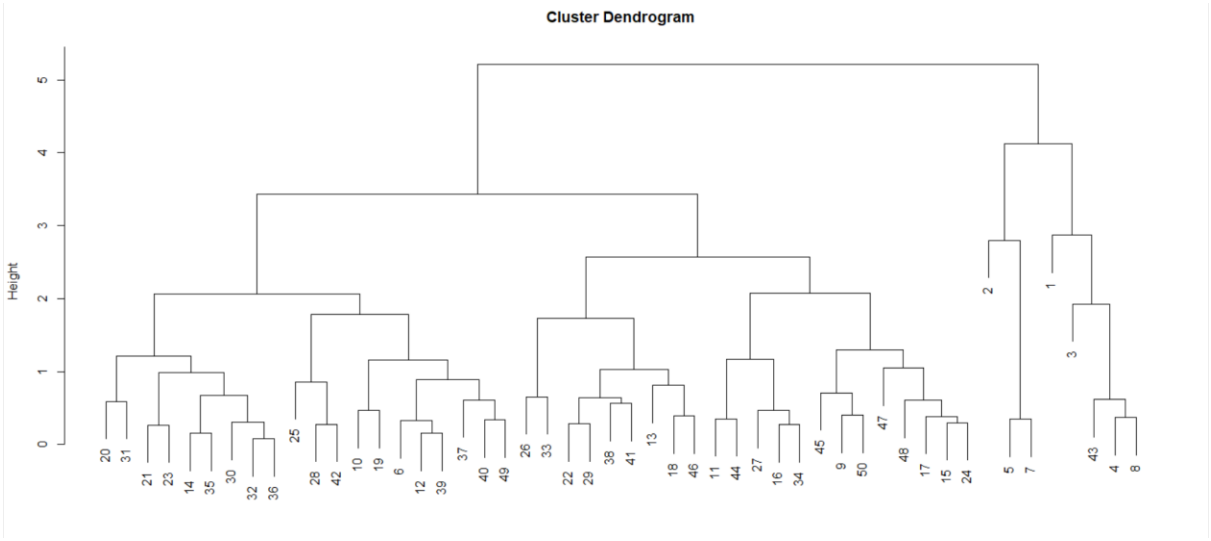

Figure 39 Dendrogram

ขั้นตอนที่ 3 : จากผลลัพธ์ที่ได้ใน dendrogram จึงจัดสินใจด้วยการเลือกจำนวนกลุ่มทั้งหมด 3 กลุ่ม โดยตัดกลุ่มที่ระดับความสูง 3.75

```
plot(Hierar_cl_p)
abline(h=3.75,col = "green")
```

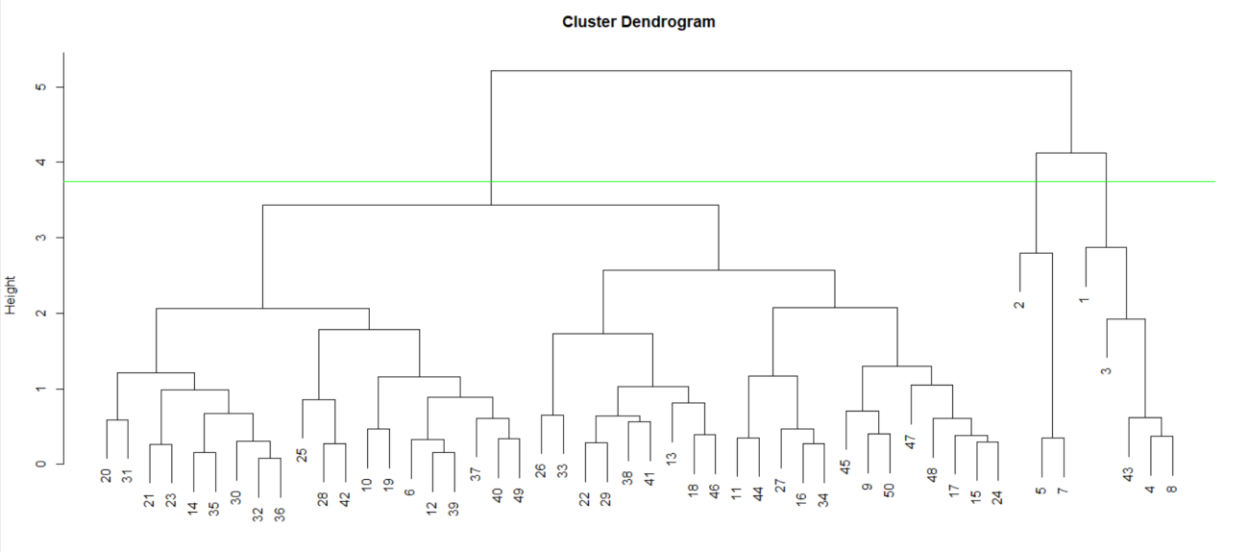

Figure 40 จุดตัดสินใจสำหรับการสร้างกลุ่ม

ขั้นตอนที่ 4 : เราจึงทำการแบ่งกลุ่มเท่ากับ 3 กลุ่มโดยที่ใช้ fit = 3 เพื่อแสดงผลของแต่ละกลุ่มจากนั้น plot ด้วยคำสั่ง plot

พบแบ่งกลุ่มได้ 3 กลุ่มได้แก่ 5,3 และ 42

```
fit<- cutree(Hierar_cl_p,k=3)
fit
plot(Hierar_cl_p)
table(fit)
rect.hclust(Hierar_cl_p,k=3,border = "green")
```

```
fit
 1  2  3
 5  3 42
```

Figure 41 โค้ดและจำนวนของการจัดกลุ่ม

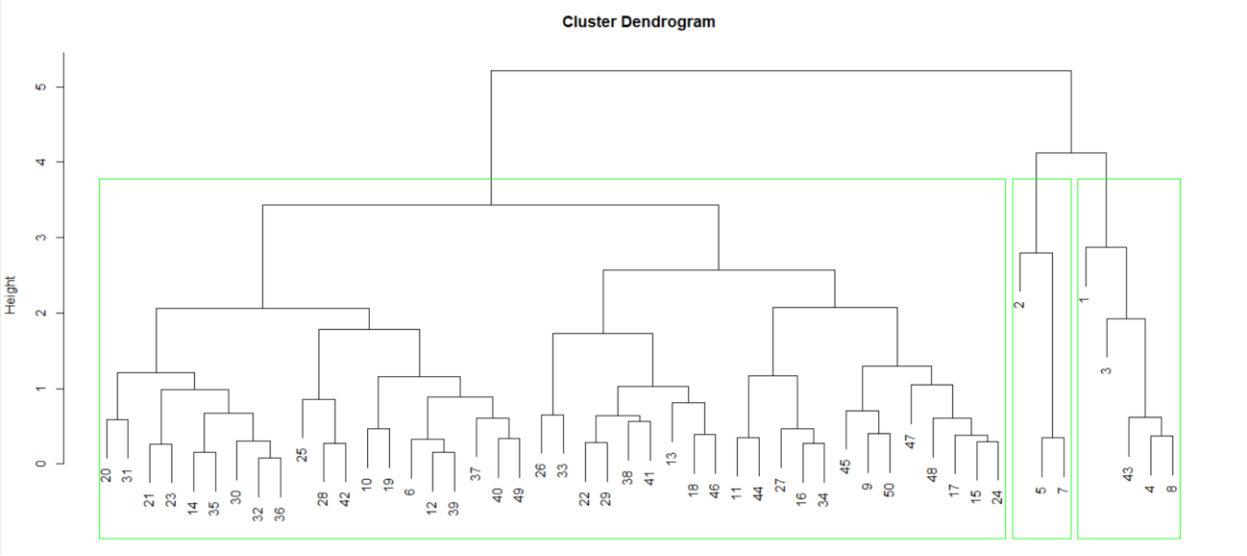

Figure 42 Dendrogram ที่จัดกลุ่มเรียบร้อยแล้ว

สรุปผลจากขั้นตอนทั้งหมดสามารถสรุปได้ว่าการทำ Hierarchical Clustering ด้วยวิธีการ Euclidean distance จากข้อมูลกรณีทำ PCA จำนวน 50 ข้อมูลพบว่าภายใต้การตัดสินใจจากการทำ Dendrogram จึงตัดสินใจจัดกลุ่ม 3 ที่ความสูง 3.75 ประกอบไปด้วยได้แก่ 5,3 และ 42

4.6 Hierarchical Clustering ด้วย Manhattan Distance

กรณีที่ไม่ได้ทำ PCA

ขั้นตอนที่ 1 : กำหนดระยะทางที่จะใช้ด้วยและข้อมูลที่ใช้ด้วย dist(test,method = 'manhattan') จากนั้นนำไปใช้ใน hclust เพื่อทำการสร้างกลุ่มแบบ Hierarchical Clustering

```
distance_mat_m <- dist(test,method = 'manhattan')
distance_mat_m
Hierar_cl_m <- hclust(distance_mat_m)
Hierar_cl_m
```

Figure 43 คำนวณระยะห่างและจัดกลุ่ม Heirarchical

ขั้นตอนที่ 2 : ทำเช็คข้อมูลจากการรันและทำการ plot ในรูปแบบ Dendrogram

```
Call:
hclust(d = distance_mat_m)

Cluster method   : complete
Distance         : manhattan
Number of objects: 50
```

```
plot(Hierar_cl_m)
```

Figure 44 ข้อมูลการรันและการสร้างกราฟ Hierarchical

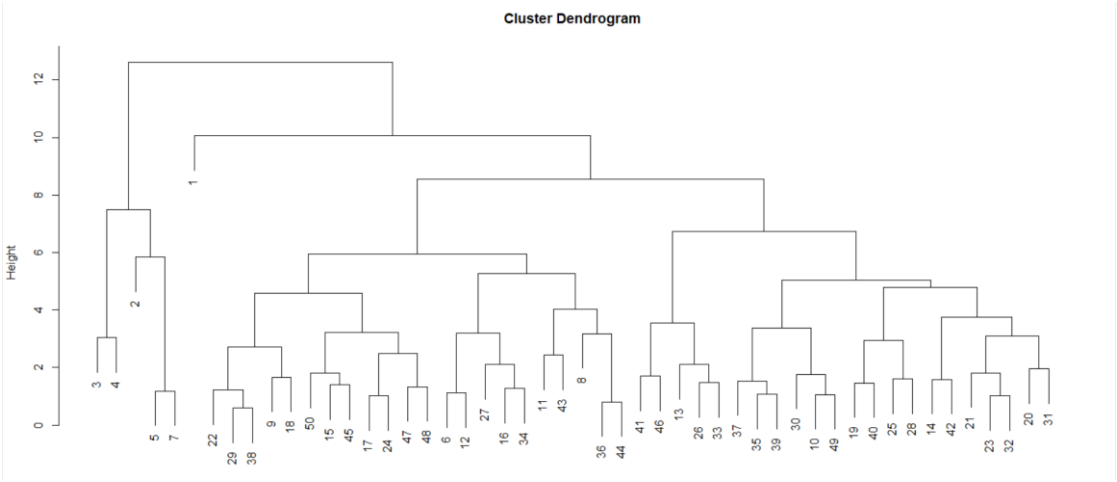

Figure 45 Dendrogram

ขั้นตอนที่ 3 : จากผลลัพธ์ที่ได้ใน dendrogram จึงจัดสินใจด้วยการเลือกจำนวนกลุ่มทั้งหมด 3 กลุ่ม โดยตัดกลุ่มที่ระดับความสูง 9

```
plot(Hierar_cl_m)
abline(h=9,col = "green")
```

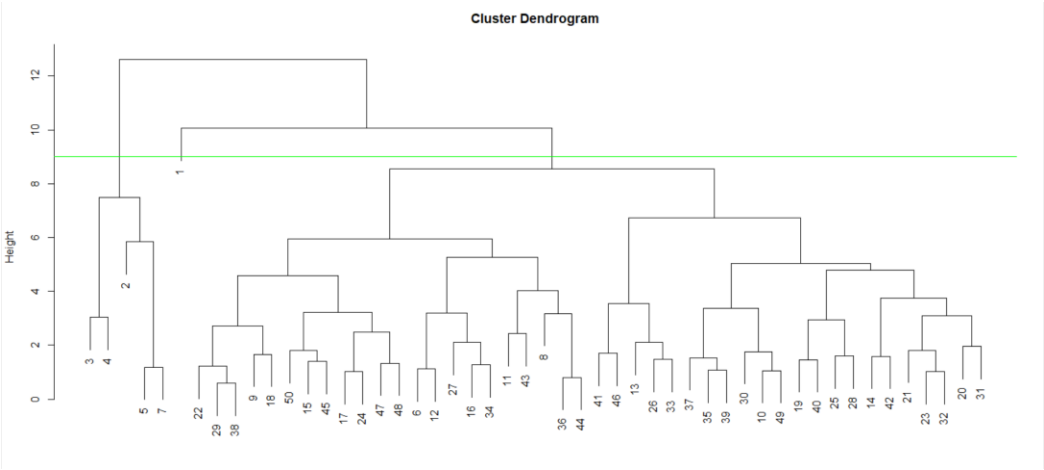

Figure 46 จุดตัดสินใจสำหรับการสร้างกลุ่ม

ขั้นตอนที่ 4 : เราจึงทำการแบ่งกลุ่มเท่ากับ 3 กลุ่มโดยที่ใช้ fit = 3 เพื่อแสดงผลของแต่ละกลุ่มจากนั้น plot ด้วยคำสั่ง plot

พบแบ่งกลุ่มได้ 3 กลุ่มได้แก่ 1,5 และ 44

```
fit<- cutree(Hierar_cl_m,k=3)
fit
plot(Hierar_cl_m)
table(fit)
rect.hclust(Hierar_cl_m,k=3,border = "green")
```

```
fit
 1  2  3
 1  5 44
```

Figure 47 โค้ดและจำนวนของการจัดกลุ่ม

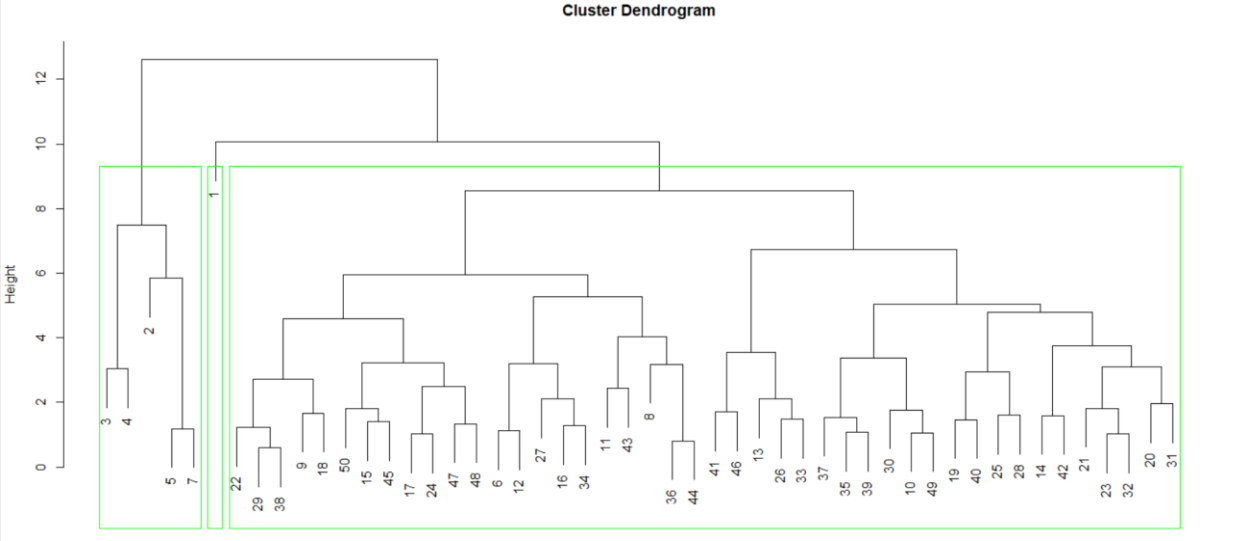

Figure 48 Dendrogram ที่จัดกลุ่มเรียบร้อยแล้ว

สรุปผลจากขั้นตอนทั้งหมดสามารถสรุปได้ว่าการทำ Hierarchical Clustering ด้วยวิธีการ Manhattan distance จากข้อมูลกรณีไม่ได้ทำ PCA จำนวน 50 ข้อมูลพบว่าภายใต้การตัดสินใจจากการทำ Dendrogram จึงตัดสินใจจัดกลุ่ม 3 ที่ความสูง 9 ประกอบไปด้วยได้แก่ 1,5 และ 44

กรณีที่ทำ PCA

ขั้นตอนที่ 1 : กำหนดระยะทางที่จะใช้ด้วยและข้อมูลที่ใช้ด้วย dist(test_pca,method = 'manhattan') จากนั้นนำไปใช้ใน hclust เพื่อทำการสร้างกลุ่มแบบ Hierarchical Clustering

```
distance_mat_pm <- dist(test_pca,method = 'manhattan')
distance_mat_pm
Hierar_cl_pm <- hclust(distance_mat_pm)
Hierar_cl_pm
```

Figure 49 คำนวณระยะห่างและจัดกลุ่ม Heirarchical

ขั้นตอนที่ 2 : ทำเช็คข้อมูลจากการรันและทำการ plot ในรูปแบบ Dendrogram

```
Call:
hclust(d = distance_mat_pm)

Cluster method   : complete
Distance         : manhattan
Number of objects: 50
```

```
plot(Hierar_cl_pm)
```

Figure 50 ข้อมูลการรันและการสร้างกราฟ Hierarchical

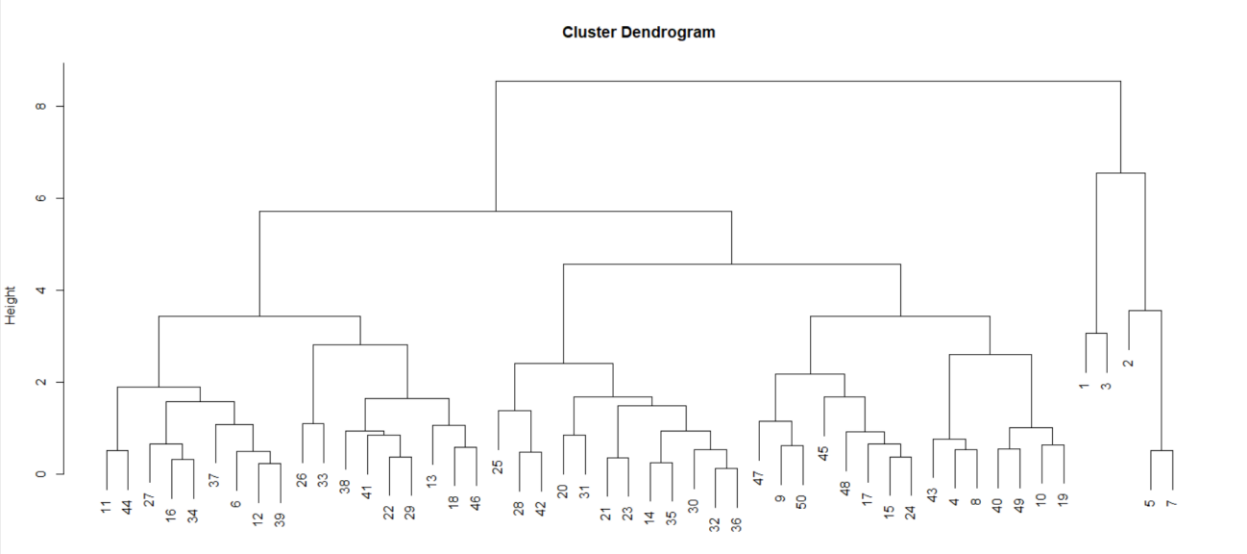

Figure 51 Dendrogram

ขั้นตอนที่ 3 : จากผลลัพธ์ที่ได้ใน dendrogram จึงจัดสินใจด้วยการเลือกจำนวนกลุ่มทั้งหมด 3 กลุ่ม โดยตัดกลุ่มที่ระดับความสูง 6

```
plot(Hierar_cl_pm)
abline(h=6,col = "green")
```

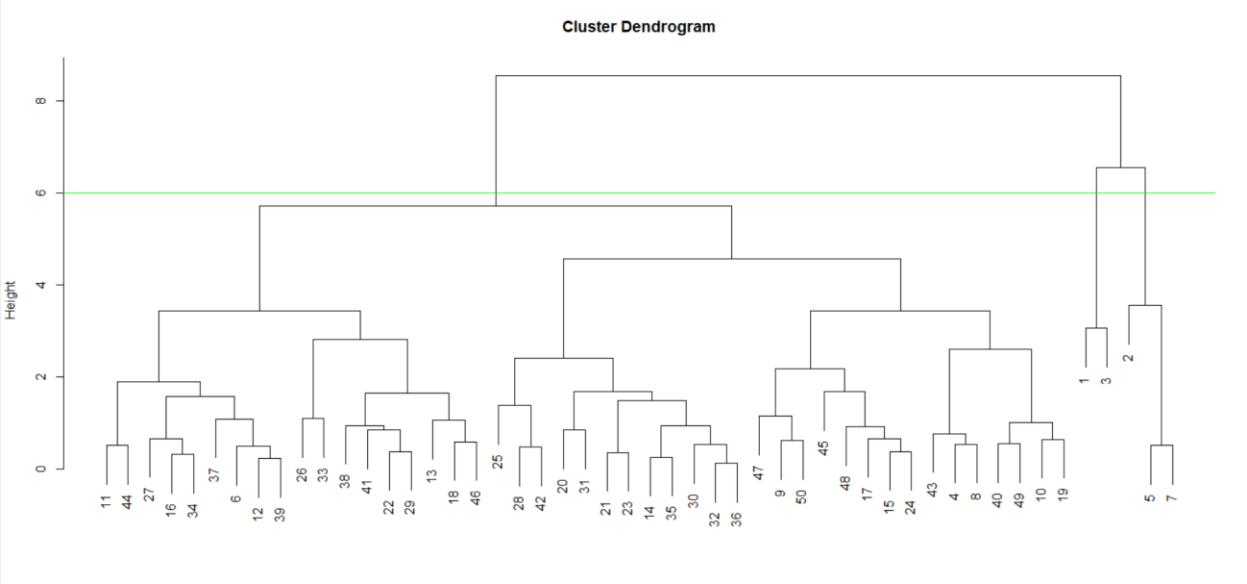

Figure 52 จุดตัดสินใจสำหรับการสร้างกลุ่ม

ขั้นตอนที่ 4 : เราจึงทำการแบ่งกลุ่มเท่ากับ 3 กลุ่มโดยที่ใช้ fit = 3 เพื่อแสดงผลของแต่ละกลุ่มจากนั้น plot ด้วยคำสั่ง plot

พบแบ่งกลุ่มได้ 3 กลุ่มได้แก่ 2,3 และ 45

```
fit<- cutree(Hierar_cl_pm,k=3)
fit
plot(Hierar_cl_pm)
table(fit)
rect.hclust(Hierar_cl_pm,k=3,border = "green")
```

```
> table(fit)
fit
 1  2  3
 2  3 45
```

Figure 53 โค้ดและจำนวนของการจัดกลุ่ม

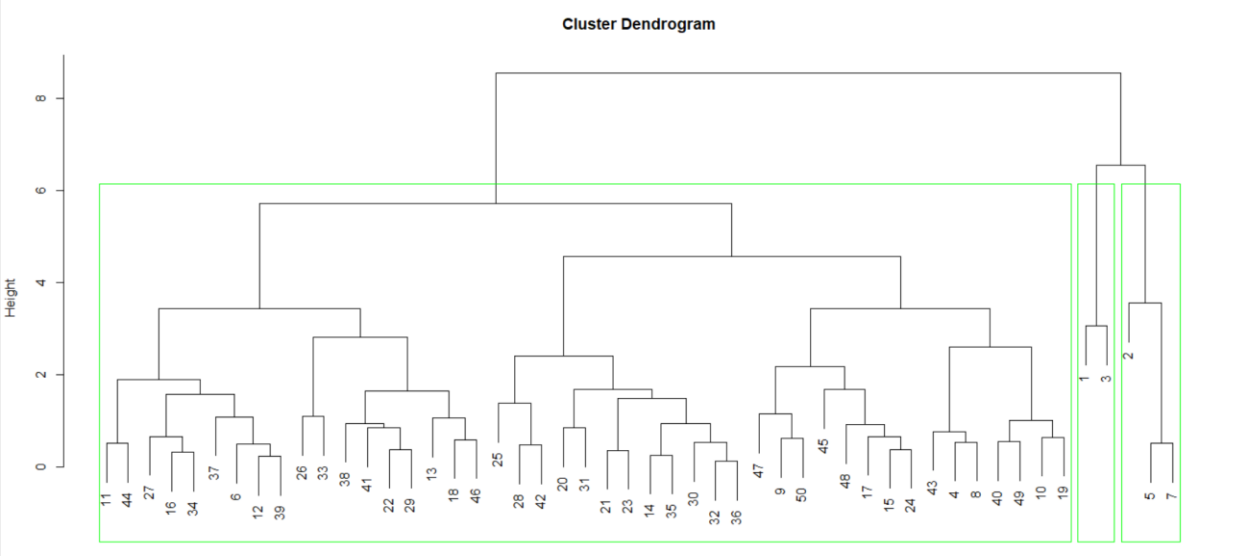

Figure 54 Dendrogram ที่จัดกลุ่มเรียบร้อยแล้ว

สรุปผลจากขั้นตอนทั้งหมดสามารถสรุปได้ว่าการทำ Hierarchical Clustering ด้วยวิธีการ Manhattan distance จากข้อมูลกรณีทำ PCA จำนวน 50 ข้อมูลพบว่าภายใต้การตัดสินใจจากการทำ Dendrogram จึงตัดสินใจจัดกลุ่ม 3 ที่ความสูง 6 ประกอบไปด้วยได้แก่ 2,3 และ 45

4.7 DBScan ด้วย Euclidean Distance

กรณีที่ไม่ได้ทำ PCA

ขั้นตอนที่ 1 กำหนดระยะห่างเป็น Euclidean distance โดยใช้คำสั่ง proxy กำหนดข้อมูลเป็นข้อมูลกรณีไม่ได้ทำ PCA และวิธีการ Euclidean หลังจากกำหนดระยะทางได้ทำการจัดกลุ่มด้วย DBScan โดยกำหนด eps =1 และ MinPts = 3

```
dist_matrix_eu <- proxy::dist(scale_data,method = "Euclidean")

Db_cl <- dbscan::dbscan(dist_matrix_eu,eps = 1, minPts = 3)
Db_cl
```

Figure 55 คำนวนระยะห่างและตั้งพารามิเตอร์

ขั้นตอนที่ 2 จากผลลัพธ์จะได้ว่าสามารถจัดกลุ่มได้ 6 กลุ่มได้แก่ 605,3,8,4,3 และ 3 ตามลำดับและพบค่า Noise ทั้งหมด 34 ค่า

```
DBSCAN clustering for 660 objects.
Parameters: eps = 1, minPts = 3
Using Euclidean distances and borderpoints = TRUE
The clustering contains 6 cluster(s) and 34 noise points.

  0   1   2   3   4   5   6
 34 605   3   8   4   3   3
```

Figure 56 ภาพรวมของ DBScan ของวิธีนี้

ขั้นตอนที่ 3 ทำการ plot เพื่อดูข้อมูลภาพรวมของแต่ละกลุ่ม

```
plot(scale_data, col = Db_cl$cluster)
```

Figure 57 พล็อตกราฟข้อมูลที่ไม่ใช่ noise

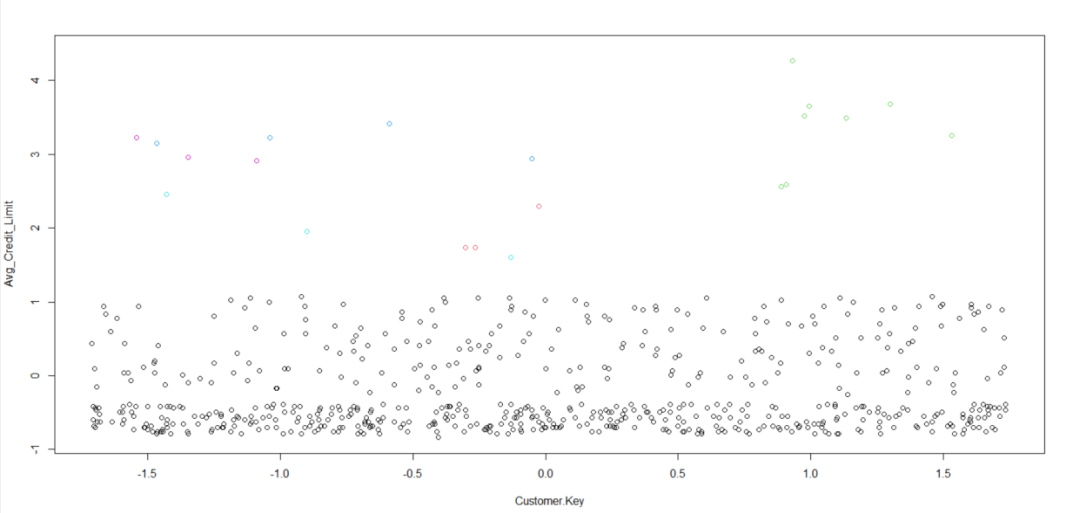

Figure 58 กราฟแสดงกลุ่มของแต่ละกลุ่ม

สรุปผลจากขั้นตอนทั้งหมดสามารถสรุปได้ว่าการทำ DBScan ด้วยวิธีการ Euclidean distance จากข้อมูลกรณีไม่ได้ทำ PCA พบว่าภายใต้การตัดสินใจจัดกลุ่มที่ eps = 1 และ MinPts = 3 จะสามารถจัดกลุ่มได้ทั้งหมด 6 กลุ่ม ได้แก่ 605,3,8,4,3 และ 3 ตามลำดับและพบค่า Noise ทั้งหมด 34 ค่า

กรณีที่ทำ PCA

ขั้นตอนที่ 1 กำหนดระยะห่างเป็น Euclidean distance โดยใช้คำสั่ง proxy กำหนดข้อมูลเป็นข้อมูลกรณีทำ PCA และวิธีการ Euclidean หลังจากกำหนดระยะทางได้ทำการจัดกลุ่มด้วย DBScan โดยกำหนด eps =1 และ MinPts = 3

```
dist_matrix_p <- proxy::dist(pca_data,method = "Euclidean")

Db_cl_p <- dbscan::dbscan(dist_matrix_p,eps = 1, minPts = 3)
Db_cl_p
```

Figure 59 คำนวนระยะห่างและตั้งพารามิเตอร์

ขั้นตอนที่ 2 จากผลลัพธ์จะได้ว่าสามารถจัดกลุ่มได้ 3 กลุ่มได้แก่ 609,46, และ 4 ตามลำดับและพบค่า Noise ทั้งหมด 1 ค่า

```
DBSCAN clustering for 660 objects.
Parameters: eps = 1, minPts = 3
Using Euclidean distances and borderpoints = TRUE
The clustering contains 3 cluster(s) and 1 noise points.

  0   1   2   3
  1 609  46   4
```

Figure 60 ภาพรวมของ DBScan ของวิธีนี้

ขั้นตอนที่ 3 ทำการ plot เพื่อดูข้อมูลภาพรวมของแต่ละกลุ่ม

```
plot(pca_data, col = Db_cl_p$cluster)
```

Figure 61 พล็อตกราฟข้อมูลที่ไม่ใช่ noise

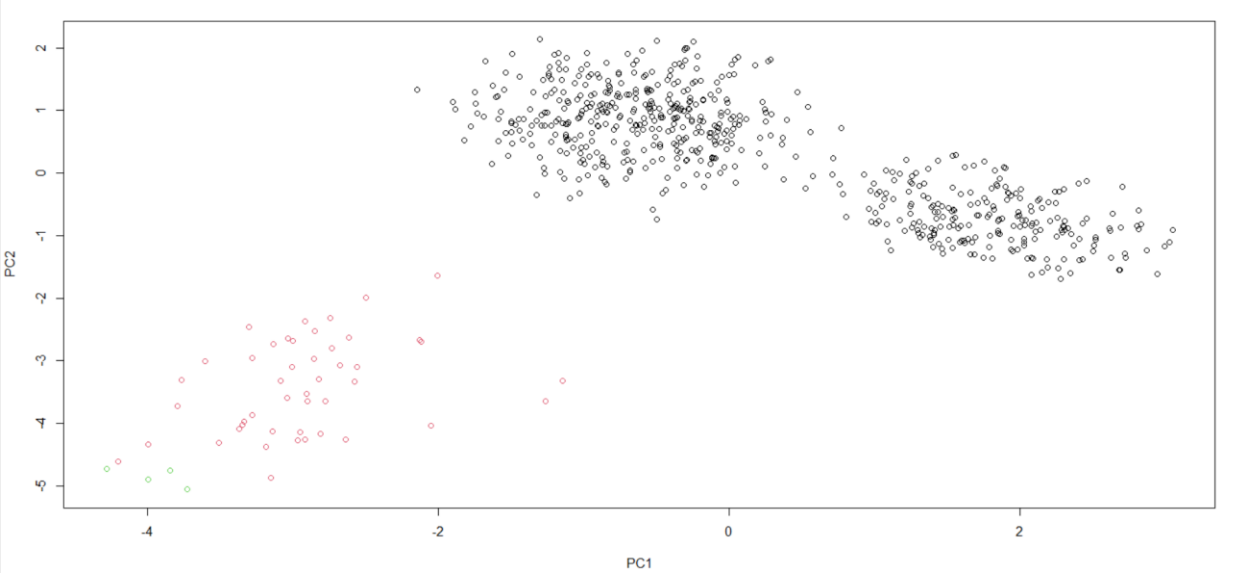

Figure 62 กราฟแสดงกลุ่มของแต่ละกลุ่ม

สรุปผลจากขั้นตอนทั้งหมดสามารถสรุปได้ว่าการทำ DBScan ด้วยวิธีการ Euclidean distance จากข้อมูลกรณีทำ PCA พบว่าภายใต้การตัดสินใจจัดกลุ่มที่ eps = 1 และ MinPts = 3 จะสามารถจัดกลุ่มได้ทั้งหมด 3 กลุ่ม ได้แก่ 609,46 และ 4 ตามลำดับและพบค่า Noise ทั้งหมด 1 ค่า

4.7 DBScan ด้วย Manhattan Distance

กรณีที่ไม่ได้ทำ PCA

ขั้นตอนที่ 1 กำหนดระยะห่างเป็น Manhattan distance โดยใช้คำสั่ง proxy กำหนดข้อมูลเป็นข้อมูลกรณีไม่ได้ทำ PCA และวิธีการ Manhattan หลังจากกำหนดระยะทางได้ทำการจัดกลุ่มด้วย DBScan โดยกำหนด eps =1 และ MinPts = 3

```
dist_matrix_m <- proxy::dist(scale_data,method = "Manhattan")

Db_cl_m <- dbscan::dbscan(dist_matrix_m,eps = 1, minPts = 3)
Db_cl_m
```

Figure 63 คำนวนระยะห่างและตั้งพารามิเตอร์

ขั้นตอนที่ 2 จากผลลัพธ์จะได้ว่าสามารถจัดกลุ่มได้ 41 กลุ่มพบค่า Noise ทั้งหมด 294 ค่า

```
DBSCAN clustering for 660 objects.
Parameters: eps = 1, minPts = 3
Using Manhattan distances and borderpoints = TRUE
The clustering contains 41 cluster(s) and 294 noise points.

  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22
294   5  12  24  43   3   4  12  13   7   3   3   5   3   4   4   5  15  25   3  28  37  11
 23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41
  8  18   6   3   3   4   3   3   7   5  10   4   4   3   4   3   3   3   3
```

Figure 64 ภาพรวมของ DBScan ของวิธีนี้

ขั้นตอนที่ 3 ทำการ plot เพื่อดูข้อมูลภาพรวมของแต่ละกลุ่ม

```
plot(scale_data, col = Db_cl_m$cluster)
```

Figure 65 พล็อตกราฟข้อมูลที่ไม่ใช่ noise

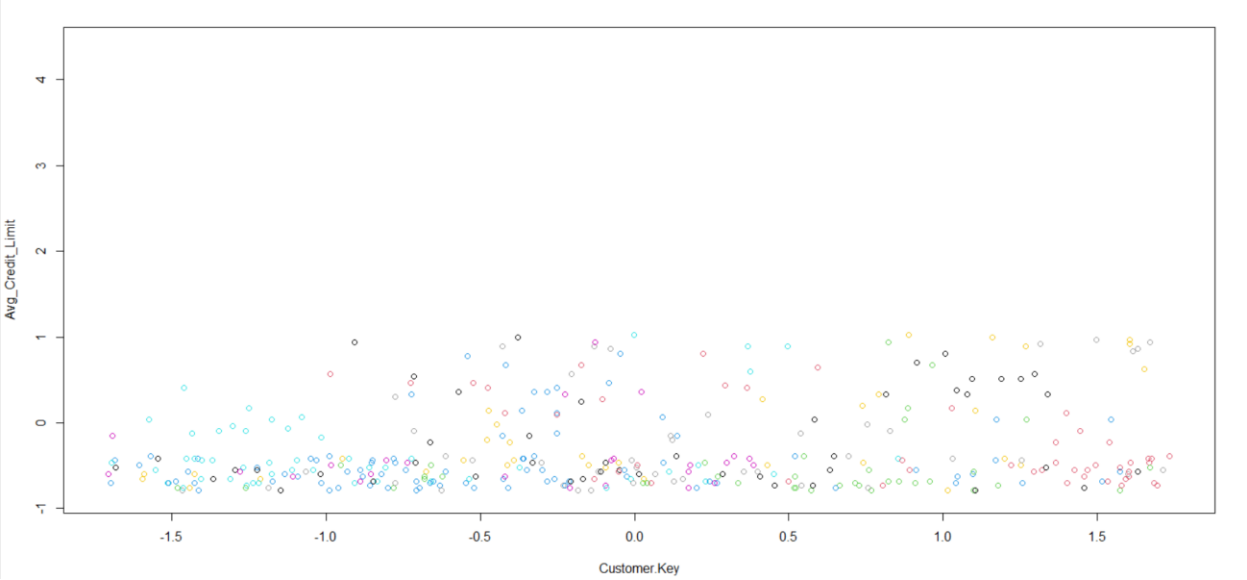

Figure 66 กราฟแสดงกลุ่มของแต่ละกลุ่ม

สรุปผลจากขั้นตอนทั้งหมดสามารถสรุปได้ว่าการทำ DBScan ด้วยวิธีการ Manhattan distance จากข้อมูลกรณีไม่ได้ทำ PCA พบว่าภายใต้การตัดสินใจจัดกลุ่มที่ eps = 1 และ MinPts = 3 จะสามารถจัดกลุ่มได้ทั้งหมด 41 และพบค่า Noise ทั้งหมด 294 ค่า

กรณีที่ทำ PCA

ขั้นตอนที่ 1 กำหนดระยะห่างเป็น Manhattan distance โดยใช้คำสั่ง proxy กำหนดข้อมูลเป็นข้อมูลกรณีทำ PCA และวิธีการ Manhattan หลังจากกำหนดระยะทางได้ทำการจัดกลุ่มด้วย DBScan โดยกำหนด eps =1 และ MinPts = 3

```
dist_matrix_pm <- proxy::dist(pca_data,method = "Manhattan")

Db_cl_pm <- dbscan::dbscan(dist_matrix_pm,eps = 1, minPts = 3)
Db_cl_pm
```

Figure 67 คำนวนระยะห่างและตั้งพารามิเตอร์

ขั้นตอนที่ 2 จากผลลัพธ์จะได้ว่าสามารถจัดกลุ่มได้ 3 กลุ่มได้แก่ 608,38, และ 3 ตามลำดับและพบค่า Noise ทั้งหมด 11 ค่า

```
DBSCAN clustering for 660 objects.
Parameters: eps = 1, minPts = 3
Using Manhattan distances and borderpoints = TRUE
The clustering contains 3 cluster(s) and 11 noise points.

  0   1   2   3
 11 608  38   3
```

Figure 68 ภาพรวมของ DBScan ของวิธีนี้

ขั้นตอนที่ 3 ทำการ plot เพื่อดูข้อมูลภาพรวมของแต่ละกลุ่ม

```
plot(pca_data, col = Db_cl_pm$cluster)
```

Figure 69 พล็อตกราฟข้อมูลที่ไม่ใช่ noise

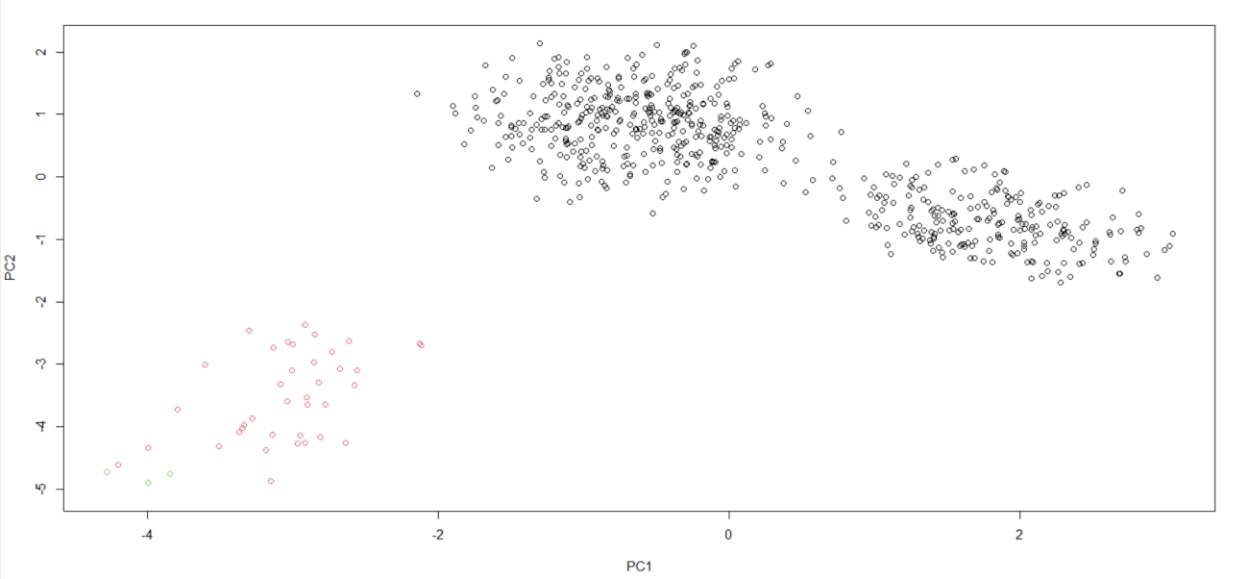

Figure 70 กราฟแสดงกลุ่มของแต่ละกลุ่ม

สรุปผลจากขั้นตอนทั้งหมดสามารถสรุปได้ว่าการทำ DBScan ด้วยวิธีการ Manhattan distance จากข้อมูลกรณีทำ PCA พบว่าภายใต้การตัดสินใจจัดกลุ่มที่ eps = 1 และ MinPts = 3 จะสามารถจัดกลุ่มได้ทั้งหมด 3 กลุ่ม ได้แก่ 608,38, และ 3 ตามลำดับและพบค่า Noise ทั้งหมด 11 ค่า

5. (Compare) การเปรียบเทียบ

ในการเปรียบจะเปรียบเทียบด้วยวิธีการ Visualization หรือจากการพล็อตกราฟโดยประกอบไปด้วยดังนี้

5.1 K-means ด้วย Euclidean Distance ทั้งกรณีที่ไม่ได้ทำ PCA และกรณีที่ทำ PCA

Figure 71 เปรียบเทียบ K-means ด้วย Euclidean Distance ทั้งกรณีที่ไม่ได้ทำ PCA (บน) และกรณีที่ทำ PCA (ล่าง)

ในการเปรียบเทียบพบว่าทั้งกรณีที่ไม่ได้ทำ PCA และกรณีที่ทำ PCA พบว่าได้จำนวนกลุ่มที่เท่ากันคือ 3 กลุ่มและแต่สมาชิกของแต่ละกลุ่มเท่ากัน ยกเว้นกลุ่มที่ 2 กับกลุ่มที่ 3 ที่สลับจำนวนสมาชิกในกลุ่มกัน ทั้งนี้ทั้งนั้นค่า total sum of square ของกรณีที่ไม่ได้ทำ PCA น้อยกว่า total sum of square ของกรณีทำ PCA (59.8% < 69.4%)

5.2 K-means ด้วย Manhattan Distance ทั้งกรณีที่ไม่ได้ทำ PCA และกรณีที่ทำ PCA

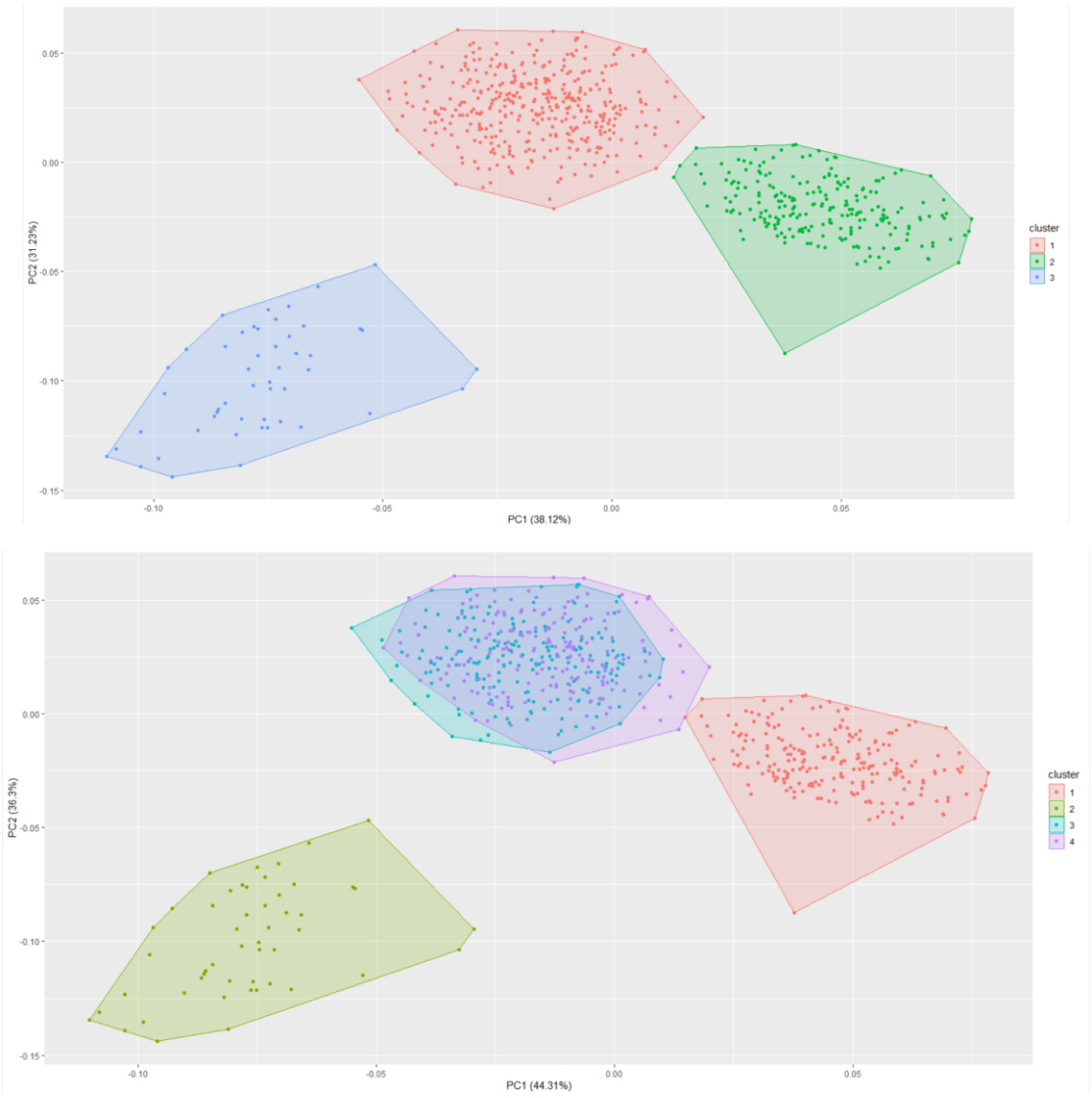

Figure 72 เปรียบเทียบ K-means ด้วย Manhattan Distance ทั้งกรณีที่ไม่ได้ทำ PCA (บน) และกรณีที่ทำ PCA (ล่าง)

ในการเปรียบเทียบพบว่าทั้งกรณีที่ไม่ได้ทำ PCA และกรณีที่ทำ PCA พบว่าได้จำนวนกลุ่มที่แตกต่างกันคือ 3 กลุ่มสำหรับกรณีไม่ได้ทำ PCA 4 กลุ่มสำหรับกรณที่ทำ PCA ทำให้สมาชิกของแต่ละกลุ่มนั้นแตกต่างกัน ทั้งนี้ทั้งนั้นค่า total sum of square ของกรณีที่ไม่ได้ทำ PCA น้อยกว่า total sum of square ของกรณีทำ PCA (59.8% < 77.7%)

5.3 Hierarchical Clustering ด้วย Euclidean Distance ทั้งกรณีที่ไม่ได้ทำ PCA และกรณีที่ทำ PCA

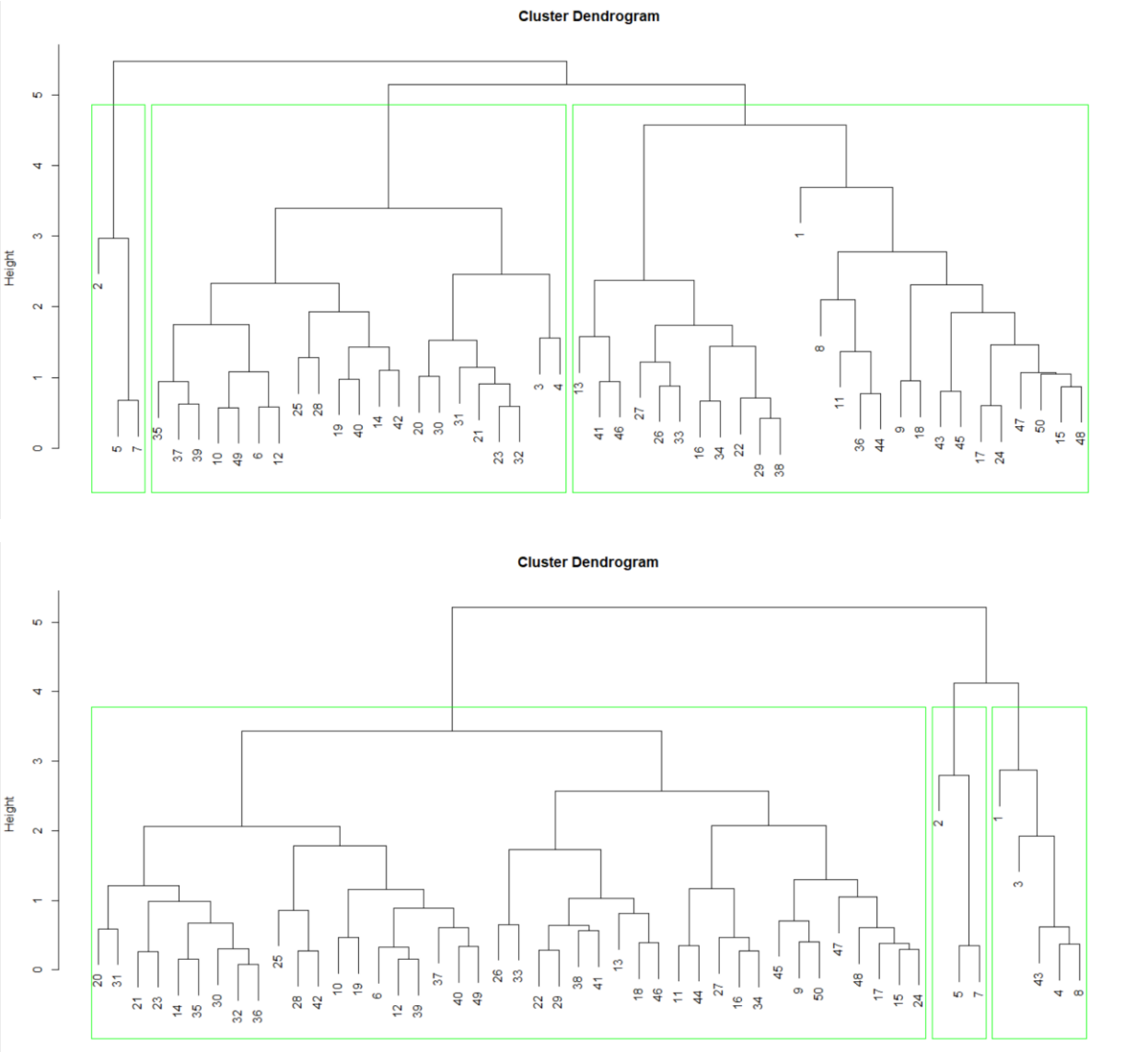

Figure 73 เปรียบเทียบ Hierarchical Clustering ด้วย Euclidean Distance ทั้งกรณีที่ไม่ได้ทำ PCA (บน) และกรณีที่ทำ PCA (ล่าง)

ในการเปรียบเทียบพบว่าทั้งกรณีที่ไม่ได้ทำ PCA และกรณีที่ทำ PCA ด้วยภายใต้การตัดสินใจเลือกจำนวนกลุ่มเท่ากับ 3 พบว่า ทำให้สมาชิกของแต่ละกลุ่มนั้นแตกต่างกัน ทั้งนี้ทั้งนั้นค่าระยะห่างระหว่างข้อมูลของกรณีที่ไม่ได้ทำ PCA มากกว่า ค่าระยะห่างระหว่างข้อมูลของกรณีที่ทำ PCA (4.75 > 3.75)

5.4 Hierarchical Clustering ด้วย Manhattan Distance ทั้งกรณีที่ไม่ได้ทำ PCA และกรณีที่ทำ PCA

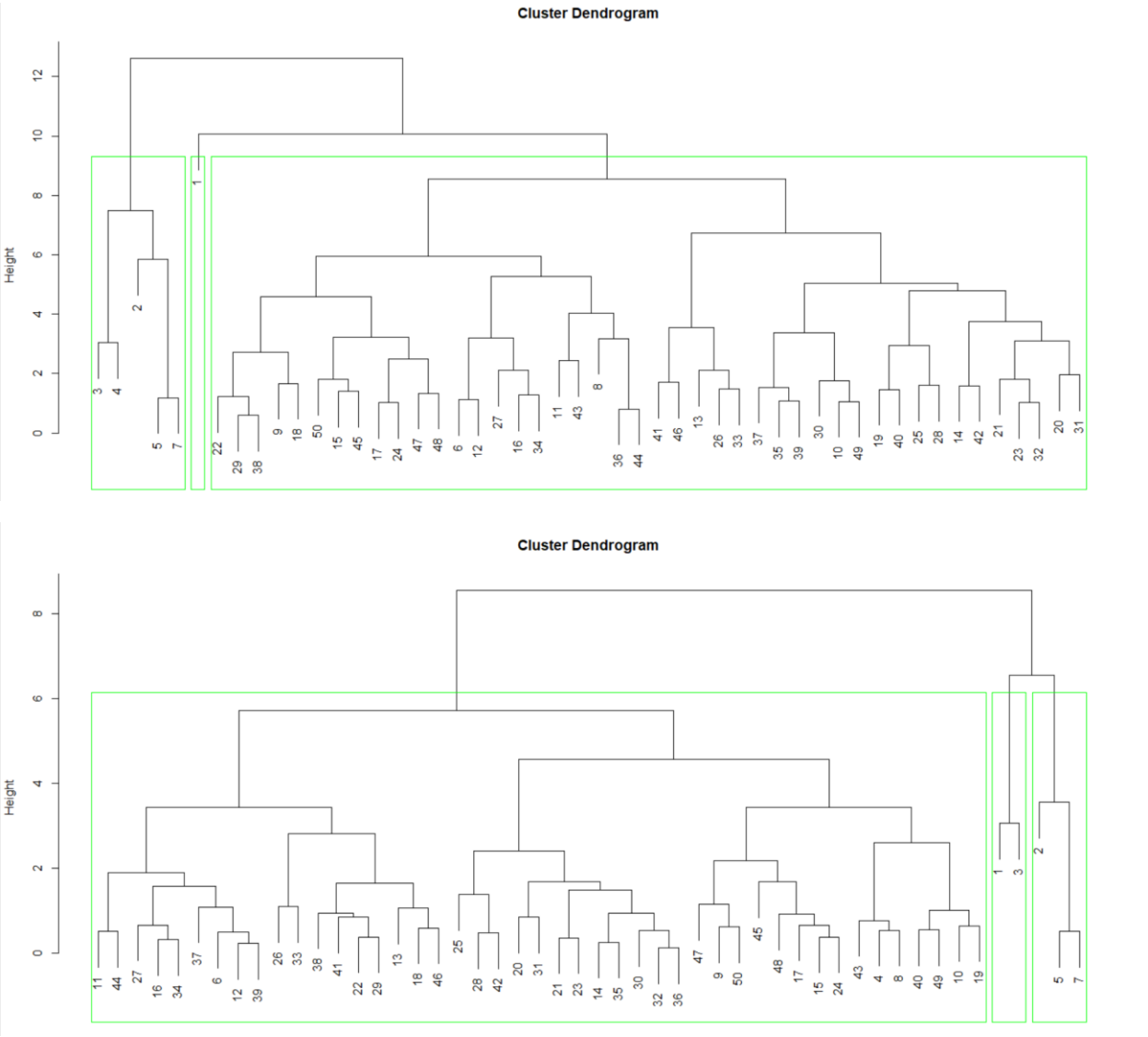

Figure 74 เปรียบเทียบ Hierarchical Clustering ด้วย Manhattan Distance ทั้งกรณีที่ไม่ได้ทำ PCA (บน) และกรณีที่ทำ PCA (ล่าง)

ในการเปรียบเทียบพบว่าทั้งกรณีที่ไม่ได้ทำ PCA และกรณีที่ทำ PCA ด้วยภายใต้การตัดสินใจเลือกจำนวนกลุ่มเท่ากับ 3 พบว่า ทำให้สมาชิกของแต่ละกลุ่มนั้นแตกต่างกัน ทั้งนี้ทั้งนั้นค่าระยะห่างระหว่างข้อมูลของกรณีที่ไม่ได้ทำ PCA มากกว่า ค่าระยะห่างระหว่างข้อมูลของกรณีที่ทำ PCA (9 > 6)

5.5 DBScan ด้วย Euclidean Distance ทั้งกรณีที่ไม่ได้ทำ PCA และกรณีที่ทำ PCA

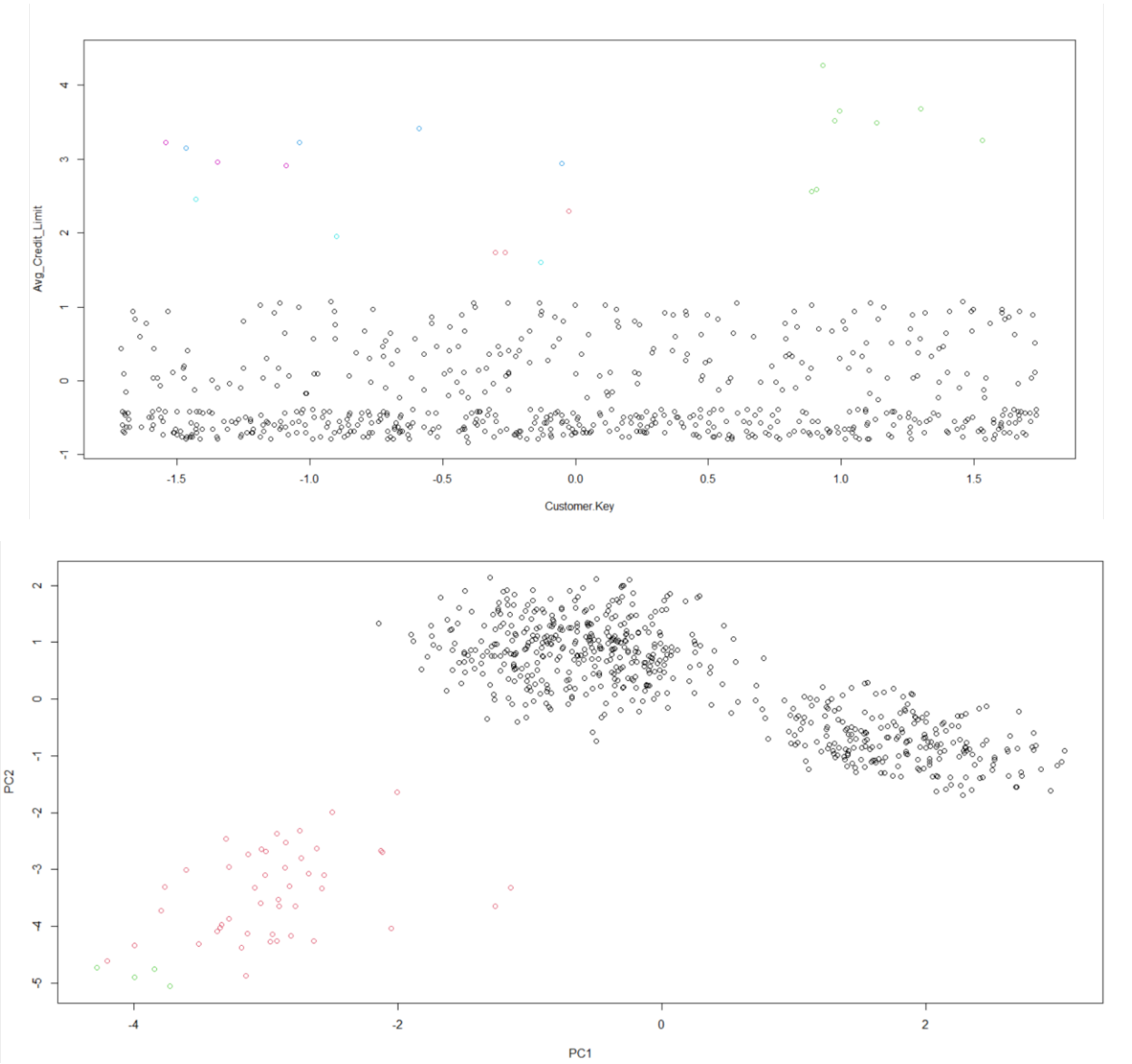

Figure 75 เปรียบเทียบ DBScan ด้วย Euclidean Distance ทั้งกรณีที่ไม่ได้ทำ PCA (บน) และกรณีที่ทำ PCA (ล่าง)

ในการเปรียบเทียบพบว่าทั้งกรณีที่ไม่ได้ทำ PCA และกรณีที่ทำ PCA ด้วยภายใต้การตัดสินใจเลือก eps=1 และ MinPts = 3 พบว่าทำให้สมาชิกของแต่ละกลุ่มนั้นแตกต่างกัน ทั้งนี้ทั้งนั้นจำนวณกลุ่มข้อมูลของกรณีที่ไม่ได้ทำ PCA มากกว่า จำนวณกลุ่มข้อมูลของกรณีที่ทำ PCA (6 > 3) และจำนวน Noise ของกรณีที่ไม่ได้ทำ PCA มากกว่า จำนวณ Noise ของกรณีที่ทำ PCA (34 > 1)

5.6 DBScan ด้วย Manhattan Distance ทั้งกรณีที่ไม่ได้ทำ PCA และกรณีที่ทำ PCA

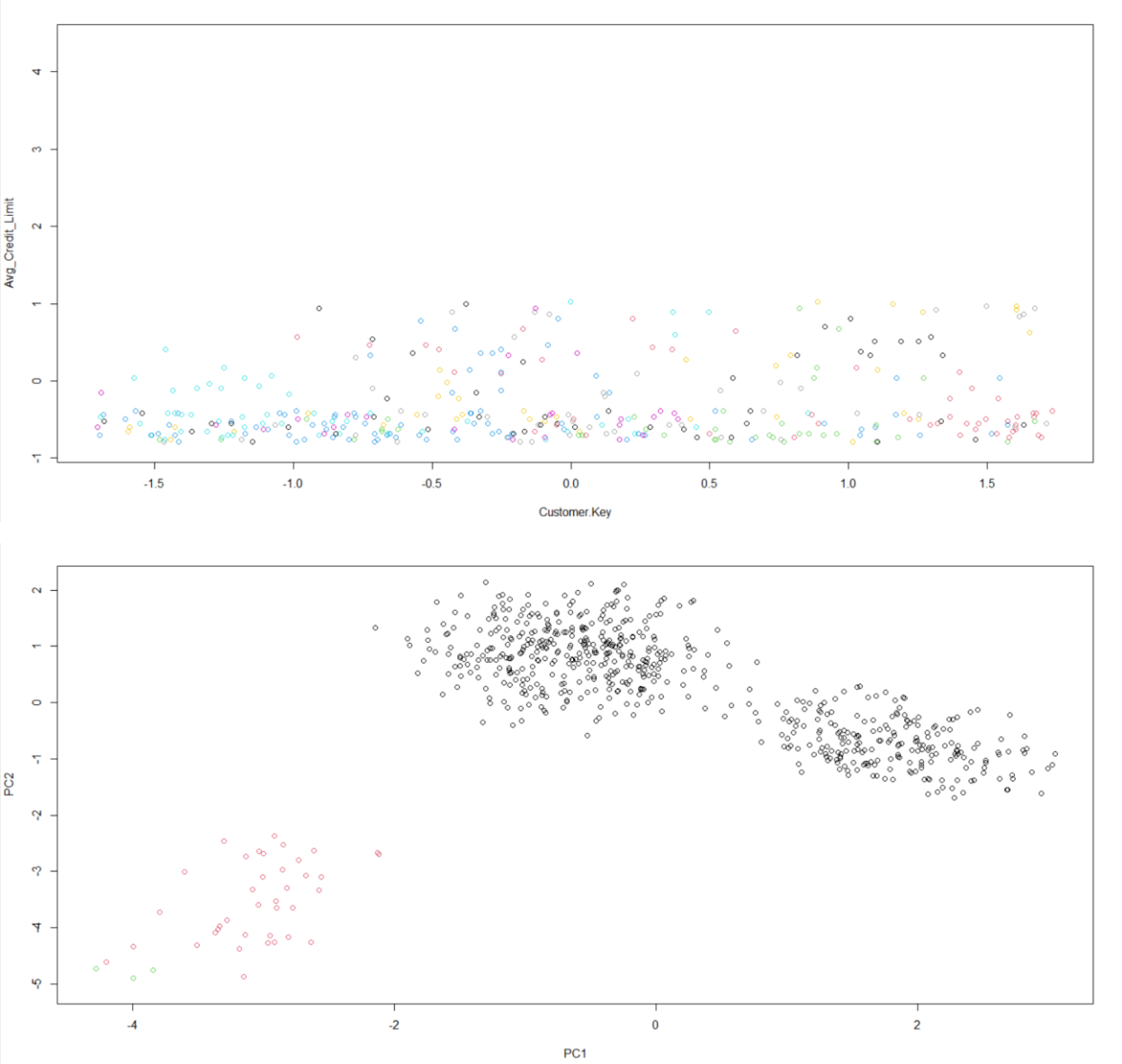

Figure 76 เปรียบเทียบ DBScan ด้วย Manhattan Distance ทั้งกรณีที่ไม่ได้ทำ PCA (บน) และกรณีที่ทำ PCA (ล่าง)

ในการเปรียบเทียบพบว่าทั้งกรณีที่ไม่ได้ทำ PCA และกรณีที่ทำ PCA ด้วยภายใต้การตัดสินใจเลือก eps=1 และ MinPts = 3 พบว่าทำให้สมาชิกของแต่ละกลุ่มนั้นแตกต่างกัน ทั้งนี้ทั้งนั้นจำนวนกลุ่มข้อมูลของกรณีที่ไม่ได้ทำ PCA มากกว่า จำนวนกลุ่มข้อมูลของกรณีที่ทำ PCA (41 > 3) และจำนวน Noise ของกรณีที่ไม่ได้ทำ PCA มากกว่า จำนวณ Noise ของกรณีที่ทำ PCA (294 > 11)

อ้างอิง

geeksforgeeks.org. (2020). **DBScan Clustering in R Programming.** สืบค้นเมื่อ 27 กันยายน 2566 ,จาก https://www.geeksforgeeks.org/dbscan-clustering-in-r-programming/

geeksforgeeks.org. (2020). **K-Means Clustering in R Programming.** สืบค้นเมื่อ 27 กันยายน 2566 , จาก https://www.geeksforgeeks.org/k-means-clustering-in-r-programming/

geeksforgeeks.org. (2021). **K Hierarchical Clustering in R Programming.** สืบค้นเมื่อ 27 กันยายน 2566, จาก https://www.geeksforgeeks.org/hierarchical-clustering-in-r-programming/

Kaggle.(2021). **Credit Card Customer Data.** สืบค้นเมื่อ 27 กันยายน 2566,จาก https://www.kaggle.com/datasets/aryashah2k/credit-card-customer-data?fbclid=IwAR2KX_qWGQolEKHFh9Yl0RqWaoEseuSly9HSfZAE4GOu14m7cA1stz29GbQ

Mathematics Learning Support Centre. (2007). **Cluster Analysis**. สืบค้นเมื่อ 27 กันยายน 2566 , จาก https://www.statstutor.ac.uk/resources/uploaded/clusteranalysis.pdf