



โครงการ  
การเรียนการสอนเพิ่มเสริมประสบการณ์

ชื่อโครงการ เอ็มโอเอฟเพื่อปรับปรุงขั้นตอนวิธีการเกาะกลุ่ม

MOF to improve clustering algorithms

ชื่อนิสิต นายจิรภัทร ชัยบุตร

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์ สาขาวิชาคณิตศาสตร์

ปีการศึกษา 2565

คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ເຄມໂອເອີເພື່ອປັບປຸງຂັ້ນຕອນວິຊີກາຮເກະກລຸ່ມ

นายຈົກສົງ  
ນາຍຈົກສົງ

ໂຄຮງຈານນີ້ເປັນສ່ວນໜຶ່ງຂອງກາຮສົງ<sup>ກາຮສົງ</sup> ທີ່  
ສາຂາວິຊາຄົນິຕະສາສົງ ກາຄວິຊາຄົນິຕະສາສົງ ແລະ ວິທາກາຮຄອມພິວເຕົວ  
ຄະນະວິທາກາສົງ ຈຸ່າລາງກຣນົມໄຫວິທາລ້ຍ  
ປີກາຮສົງ 2565  
ລືຂສີທີ່ຂອງຈຸ່າລາງກຣນົມໄຫວິທາລ້ຍ

MOF to improve clustering algorithms

Jiraphat Chaiyabut

A Project Submitted in Partial Fulfillment of the Requirements  
for the Degree of Bachelor of Science Program in Mathematics

Department of Mathematics and Computer Science

Faculty of Science

Chulalongkorn University

Academic Year 2022

Copyright of Chulalongkorn University

หัวข้อโครงการ เอกม็อกอีฟเพื่อปรับปรุงขั้นตอนวิธีการเกาเกล่อม  
โดย นายจิรภัทร ชัยบุตร  
สาขาวิชา คณิตศาสตร์  
อาจารย์ที่ปรึกษาโครงการหลัก รองศาสตราจารย์ ดร.กรุง สินอวิริมย์สรายุ

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับ  
โครงการฉบับนี้เป็นส่วนหนึ่ง ของการศึกษาตามหลักสูตรปริญญาบัณฑิต ในรายวิชา 2301499 โครงการ  
วิทยาศาสตร์ (Senior Project)

หัวหน้าภาควิชาคณิตศาสตร์

(ผู้ช่วยศาสตราจารย์ ดร.อธิปัตย์ ธรรมรงค์สุลักษณ์) และวิทยาการคอมพิวเตอร์

คณะกรรมการสอบโครงการ

อาจารย์ที่ปรึกษาโครงการหลัก

(รองศาสตราจารย์ ดร.กรุง สินอวิริมย์สรายุ)

กรรมการ

(รองศาสตราจารย์ ดร.วิชาญ ลีวิกรณ์ติยุติกุล )

กรรมการ

(รองศาสตราจารย์ ดร.ศจี เพียรสกุล)

จิรภัทร ชัยบุตร : เอมโอดอฟเพื่อปรับปรุงขั้นตอนวิธีการแกะกลุ่ม

อาจารย์ที่ปรึกษาโครงการหลัก : รศ.ดร. กรุง สินอภิรมย์สรายุ, 109 หน้า

การตรวจหาและกำจัดข้อมูลที่ผิดปกติเป็นขั้นตอนสำคัญในการเตรียมข้อมูล การpubข้อมูลผิดปกติเป็นปัญหาที่พบบ่อยและต้องใช้เทคนิคและวิธีต่างๆ เพื่อตรวจหาข้อมูลเหล่านั้นตามลักษณะหรือโครงสร้างข้อมูล ข้อมูลผิดปกติมีผลต่อความแม่นยำในการวิเคราะห์ข้อมูล ตัวอย่างหนึ่งคือการวิเคราะห์การแกะกลุ่มเป็นเทคนิคการวิเคราะห์ข้อมูลโดยการรวมกลุ่มข้อมูลที่มีรูปแบบหรือโครงสร้างเข้ากัน แต่การรวมกลุ่มดังกล่าวจะได้รับผลกระทบจากข้อมูลผิดปกติ ดังนั้นก่อนการแกะกลุ่ม ผู้วิเคราะห์จำเป็นต้องใช้เทคนิคเพื่อกำจัดข้อมูลที่ผิดปกติ

ในโครงการนี้เราใช้ขั้นตอนวิธีเอ็มโอดอฟสำหรับตรวจหาข้อมูลผิดปกติ ก่อนส่งให้ขั้นตอนวิธีการแกะกลุ่ม หลักการนี้เพิ่มประสิทธิภาพของชิลูอ์ตบันชุดข้อมูลที่สร้างขึ้นที่มีข้อมูลผิดปกติอยู่

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์ ลายมือชื่อนิสิต **จิรภัทร ชัยบุตร**

สาขาวิชา คณิตศาสตร์

ลายมือชื่อ อาจารย์ที่ปรึกษาโครงการหลัก **กรุง**

ปีการศึกษา 2565

# 6234307323: MAJOR MATHEMATICS

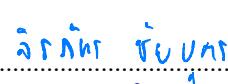
KEYWORDS : clustering algorithm / data analytics / mass-ratio variance algorithm

Jiraphat chaiyabut: MOF to improve clustering algorithms

ADVISOR : Krung Sinapiromsaran , Ph.D., 109 pp

Detecting and eliminating abnormal data is a crucial step in data preparation step. Anomalous data is a common issue that necessitates various techniques and methods for detecting them, depending on the nature and structure of the data. Such data can impact the accuracy of data analysis. Clustering, a data analytic technique that groups data to uncover hidden patterns or structures, is affected by data anomalies. Before clustering the data, it is essential to employ techniques to remove abnormal ones.

In this project, we combine the mass-ratio variance based outlier factor algorithm for anomaly detection with a clustering algorithm. This approach increases the performance for synthesized datasets having some anomalies.

Department : Mathematics and Computer Science Student's Signature ..... 

Field of Study : Mathematics Advisor's Signature ..... 

Academic Year : 2022

## กิตติกรรมประกาศ

โครงการนี้มีหัวข้อเรื่อง “เอ็มโซโซฟเพื่อปรับปรุงขั้นตอนวิธีการแก้กลุ่ม” ได้รับการสนับสนุนโดยรองศาสตราจารย์ ดร.กรุง สินอภิรมย์สรายุ อาจารย์ที่ปรึกษาโครงการได้ให้คำแนะนำ และข้อเสนอแนะต่างๆ เพื่อเป็นประโยชน์ต่อโครงการนี้ จนทำให้โครงการนี้สำเร็จลุล่วงไปด้วยดี ข้าพเจ้าขอขอบพระคุณเป็นอย่างยิ่งสำหรับความช่วยเหลือในทุกด้าน

จิรภัทร ชัยบุตร

# สารบัญ

หน้า

บทคัดย่อภาษาไทย

๔

บทคัดย่อภาษาอังกฤษ

๕

กิตติกรรมประกาศ

๗

สารบัญ

๘

สารบัญตาราง

๙

สารบัญภาพ

๑๒

บทที่ 1 บทนำ

1

1.1 ความเป็นมาและความสำคัญ

1

บทที่ 2 แนวคิด ทฤษฎี และงานวิจัยที่เกี่ยวข้อง

4

2.1 แนวคิดและทฤษฎีที่เกี่ยวข้อง

4

2.2 เทคโนโลยีที่เกี่ยวข้อง

12

2.3 งานวิจัยที่เกี่ยวข้อง

14

บทที่ 3 วิธีการดำเนินงาน

16

3.1 ประชากรและกลุ่มตัวอย่าง

16

3.2 เครื่องมือที่ใช้ในการจัดกลุ่มข้อมูล

16

3.3 การเก็บรวบรวมข้อมูล

16

3.4 การวิเคราะห์ข้อมูลและการนำเสนอ	17
3.5 เทคนิคที่นำมาใช้ในการจัดกลุ่มและวิเคราะห์ข้อมูล	17
บทที่ 4 ผลการวิเคราะห์	19
4.1 ผลการวิเคราะห์กับข้อมูลสังเคราะห์	19
4.2 ผลการวิเคราะห์ข้อมูล spotify	34
บทที่ 5 ข้อสรุปและข้อเสนอแนะ	56
5.1 สรุปผลการวิเคราะห์	56
5.2 ข้อเสนอแนะ	61
รายการอ้างอิง	62
ภาคผนวก ก แบบเสนอหัวข้อโครงการ รายวิชา 2301399 Project Proposal ปีการศึกษา 2565	65
ภาคผนวก ข โปรแกรมและตัวอย่างรหัสโปรแกรม	69
ประวัติผู้จัดทำโครงการ	110

## สารบัญตาราง

หน้า

ตารางที่ 4.1.1 แสดงคะแนนชิลูเอต ก่อนกำจัดข้อมูลผิดปกติและหลังกำจัดข้อมูลผิดปกติ ของข้อมูลรูปร่างวงกลมซ้อนสองวง	20
ตารางที่ 4.1.2 แสดงคะแนนชิลูเอตก่อนกำจัดข้อมูลผิดปกติและหลังกำจัดข้อมูลผิดปกติ ของข้อมูลรูปร่างพระจันทร์ซ้อนสองวง	22
ตารางที่ 4.1.3 แสดงคะแนนชิลูเอตก่อนกำจัดข้อมูลผิดปกติและหลังกำจัดข้อมูลผิดปกติของข้อมูล รูปร่างกลุ่มข้อมูลกระจายสามากลุ่ม	23
ตารางที่ 4.1.4 แสดงคะแนนชิลูเอตก่อนกำจัดข้อมูลผิดปกติและหลังกำจัดข้อมูลผิดปกติของ ข้อมูลรูปร่างกลุ่มข้อมูลที่เรียงเป็นเส้นสามาเส้น	25
ตารางที่ 4.1.5 แสดงคะแนนชิลูเอตก่อนกำจัดข้อมูลผิดปกติและหลังกำจัดข้อมูลผิดปกติของ ข้อมูลรูปร่างกลุ่มข้อมูลที่กระจายเป็นกลุ่มใหญ่	26
ตารางที่ 4.1.6 แสดงคะแนนชิลูเอตรูปร่างวงกลมซ้อนสองวงของวิธี Agglomerative ก่อนกำจัดข้อมูลผิดปกติและหลังกำจัดข้อมูลผิดปกติ	28
ตารางที่ 4.1.7 แสดงคะแนนชิลูเอตจากการจัดกลุ่มข้อมูลรูปร่างพระจันทร์ซ้อนสองวงโดย วิธี Agglomerative ก่อนกำจัดข้อมูลผิดปกติและหลังกำจัดข้อมูลผิดปกติ	29
ตารางที่ 4.1.8 แสดงคะแนนชิลูเอตจากการจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลกระจายสามากลุ่ม โดยวิธี Agglomerative ก่อนกำจัดข้อมูลผิดปกติและหลังกำจัดข้อมูลผิดปกติ	31
ตารางที่ 4.1.9 แสดงคะแนนชิลูเอต จากการจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลที่เรียงเป็นเส้นสามาเส้น โดยวิธี Agglomerative ก่อนกำจัดข้อมูลผิดปกติและหลังกำจัดข้อมูลผิดปกติ	32
ตารางที่ 4.1.10 แสดงคะแนนชิลูเอต จากการจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลที่กระจายเป็นกลุ่มใหญ่	

โดยวิธี Agglomerative ก่อนกำจัดข้อมูลผิดปกติและหลังกำจัดข้อมูลผิดปกติ	33
ตารางที่ 4.2.1 แสดงผลข้อมูลทั้งหมดของ spotify ตามลักษณะที่จะนำมาใช้	34
ตารางที่ 4.2.2 แสดงผลข้อมูลทั้งหมดของ spotify ตามลักษณะที่จะนำมาใช้ (ต่อ)	34
ตารางที่ 4.2.3 แสดงผลข้อมูลสุ่มตัวอย่างของ spotify ตามลักษณะที่จะนำมาใช้	34
ตารางที่ 4.2.4 แสดงผลข้อมูลสุ่มตัวอย่างของ spotify ตามลักษณะที่จะนำมาใช้ (ต่อ)	34
ตารางที่ 4.2.5 แสดงจำนวนข้อมูลแต่ละ genre ก่อนกำจัดข้อมูลผิดปกติ	35
ตารางที่ 4.2.6 แสดงจำนวนข้อมูลแต่ละ genre หลังกำจัดข้อมูลผิดปกติ	37
ตารางที่ 4.2.7 แสดงค่า Silhouette ของการจัดกลุ่มโดยวิธี agglomerative single linkage ก่อนและหลังกำจัดข้อมูลผิดปกติ	39
ตารางที่ 4.2.8 แสดงค่า Silhouette ของการจัดกลุ่มโดยวิธี k-mean ก่อนและหลังกำจัดข้อมูลผิดปกติ	41
ตารางที่ 4.2.9 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean บอกจำนวนข้อมูลแต่ละกลุ่ม	44
ตารางที่ 4.2.10 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยลักษณะ danceability แบ่งเป็น 5 ช่วงย่อย	45
ตารางที่ 4.2.11 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยลักษณะ energy แบ่งเป็น 5 ช่วงย่อย	46
ตารางที่ 4.2.12 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยลักษณะ loudness แบ่งเป็น 5 ช่วงย่อย	47
ตารางที่ 4.2.13 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยลักษณะ speechiness แบ่งเป็น 5 ช่วงย่อย	48
ตารางที่ 4.2.14 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยลักษณะ acousticness แบ่งเป็น 5 ช่วงย่อย	49
ตารางที่ 4.2.15 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยลักษณะ liveness แบ่งเป็น 5 ช่วงย่อย	50
ตารางที่ 4.2.16 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยลักษณะ valence แบ่งเป็น 5 ช่วงย่อย	51
ตารางที่ 4.2.17 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยลักษณะ tempo แบ่งเป็น 5 ช่วงย่อย	52
ตารางที่ 4.2.18 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยลักษณะ instrumentalness แบ่งเป็น 5 ช่วงย่อย	53

ตารางที่ 4.2.19 แสดงจำนวนข้อมูลกลุ่มที่ 0 โดยวิธี k-mean แบ่งตามประเภทของเพลง 54

ตารางที่ 4.2.19 แสดงจำนวนข้อมูลกลุ่มที่ 1 โดยวิธี k-mean แบ่งตามประเภทของเพลง 55

# สารบัญภาพ

	หน้า
ภาพที่ 1.1 รูปแบบการทำงานโปรแกรมแบบดั้งเดิม	1
ภาพที่ 1.2 รูปแบบการทำงานการเรียนรู้ด้วยเครื่องจักร	1
ภาพที่ 2.1 แผนภาพการทำงานของอัลกอริทึม MOF	15
ภาพที่ 4.1.1 การจัดกลุ่มข้อมูลรูปร่างวงกลมซ้อนสองวงโดยวิธี k-mean ก่อนการกำจัดข้อมูลผิดปกติ	19
ภาพที่ 4.1.2 การจัดกลุ่มข้อมูลรูปร่างวงกลมซ้อนสองวงโดยวิธี k-mean หลังการกำจัดข้อมูลผิดปกติ	20
ภาพที่ 4.1.3 การจัดกลุ่มข้อมูลรูปร่างรูปพระจันทร์ซ้อนสองวงโดยวิธี k-mean ก่อนกำจัดข้อมูลผิดปกติ	21
ภาพที่ 4.1.4 การจัดกลุ่มข้อมูลรูปร่างรูปพระจันทร์ซ้อนสองวงโดยวิธี k-mean หลังกำจัดข้อมูลผิดปกติ	21
ภาพที่ 4.1.5 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลกระจายสาม	
กลุ่มโดยวิธี k-mean ก่อนกำจัดข้อมูลผิดปกติ	22
ภาพที่ 4.1.6 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลกระจายสาม	
กลุ่มโดยวิธี k-mean หลังกำจัดข้อมูลผิดปกติ	23
ภาพที่ 4.1.7 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลที่เรียงเป็นเส้นสามเส้นโดยวิธี k-mean ก่อนกำจัดข้อมูลผิดปกติ	24
ภาพที่ 4.1.8 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลที่เรียงเป็นเส้นสามเส้นโดยวิธี k-mean หลังกำจัดข้อมูลผิดปกติ	
ภาพที่ 4.1.9 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลที่กระจายเป็นกลุ่มใหญ่โดยวิธี k-mean ก่อนกำจัดข้อมูลผิดปกติ	
ภาพที่ 4.1.10 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลที่กระจายเป็นกลุ่มใหญ่โดยวิธี k-mean หลังกำจัดข้อมูลผิดปกติ	25

ภาพที่ 4.1.11 การจัดกลุ่มข้อมูลรูปร่างวงกลมช้อนสองวงโดยวิธี Agglomerative ก่อนการกำจัดข้อมูลผิดปกติ	27
ภาพที่ 4.1.12 การจัดกลุ่มข้อมูลรูปร่างวงกลมช้อนสองวงโดยวิธี Agglomerative หลังการกำจัดข้อมูลผิดปกติ	27
ภาพที่ 4.1.13 การจัดกลุ่มข้อมูลรูปร่างพระจันทร์ช้อนสองวงโดยวิธี Agglomerative ก่อนการกำจัดข้อมูลผิดปกติ	28
ภาพที่ 4.1.14 การจัดกลุ่มข้อมูลรูปร่างพระจันทร์ช้อนสองวงโดยวิธี Agglomerative หลังการกำจัดข้อมูลผิดปกติ	29
ภาพที่ 4.1.15 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลกระจายสามกลุ่ม ก่อนการกำจัดข้อมูลผิดปกติ	30
ภาพที่ 4.1.16 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลกระจายสามกลุ่ม หลังการกำจัดข้อมูลผิดปกติ	30
ภาพที่ 4.1.17 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลที่เรียงเป็นเส้นสามเส้น ก่อนการกำจัดข้อมูลผิดปกติ	31
ภาพที่ 4.1.18 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลที่เรียงเป็นเส้นสามเส้น หลังการกำจัดข้อมูลผิดปกติ	32
ภาพที่ 4.1.19 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลที่กระจายเป็นกลุ่มใหญ่ก่อนการกำจัดข้อมูลผิดปกติ	33
ภาพที่ 4.1.20 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลที่กระจายเป็นกลุ่มใหญ่หลังการกำจัดข้อมูลผิดปกติ	33
ภาพที่ 4.2.1 กราฟวงกลมแสดงจำนวนข้อมูลแต่ละ Genre ก่อนกำจัดข้อมูลผิดปกติ	36
ภาพที่ 4.2.2 กราฟวงกลมแสดงจำนวนข้อมูลแต่ละ Genre หลังกำจัดข้อมูลผิดปกติ	37
ภาพที่ 4.2.3 กราฟแสดงเปรียบเทียบค่า Silhouette ของการจัดกลุ่มโดยวิธี agglomerative single linkage ก่อนและหลังกำจัดข้อมูลผิดปกติ	38
ภาพที่ 4.2.4 กราฟแสดงการจัดกลุ่มในแต่ละลักษณะข้อมูลโดยวิธี agglomerative single linkage ก่อนกำจัดข้อมูลผิดปกติกำหนดการแบ่งกลุ่ม 2 กลุ่ม	40
ภาพที่ 4.2.5 กราฟเปรียบค่า silhouette การจัดกลุ่มด้วย	

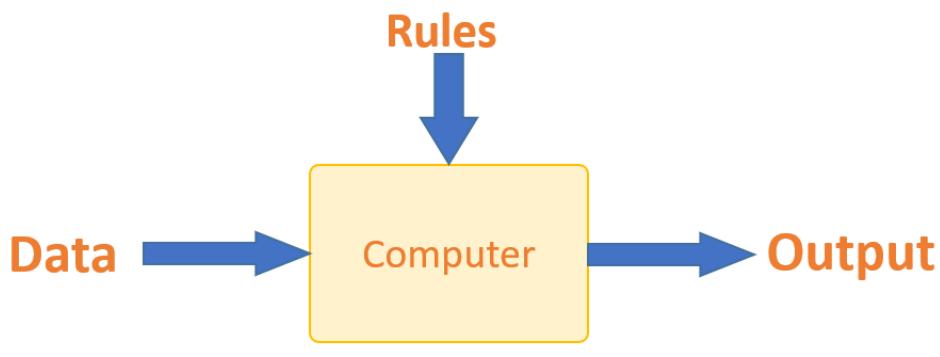
วิธี k-mean ระหว่างก่อนและหลังกำจัดข้อมูลผิดปกติ	41
ภาพที่ 4.2.6 กราฟแสดงการจัดกลุ่มในแต่ละลักษณะข้อมูลโดย	
วิธี k-mean ก่อนกำจัดข้อมูลผิดปกติกำหนดการแบ่งกลุ่ม 2 กลุ่ม	43
ภาพที่ 4.2.7 แสดงจำนวนข้อมูลแต่ละกลุ่มโดยวิธี k-mean	44
ภาพที่ 4.2.8 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean สำหรับลักษณะ danceability	45
ภาพที่ 4.2.9 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean สำหรับลักษณะ energy	46
ภาพที่ 4.2.10 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean สำหรับลักษณะ loudness	47
ภาพที่ 4.2.11 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean สำหรับลักษณะ speechiness	48
ภาพที่ 4.2.12 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean สำหรับลักษณะ acousticness	49
ภาพที่ 4.2.13 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean สำหรับลักษณะ liveness	50
ภาพที่ 4.2.14 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean สำหรับลักษณะ valence	51
ภาพที่ 4.2.15 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean สำหรับลักษณะ tempo	52
ภาพที่ 4.2.16 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean สำหรับลักษณะ intrumentalness	53
ภาพที่ 4.2.17 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean บ่งตามประเภทของเพลง	54

# บทที่ 1

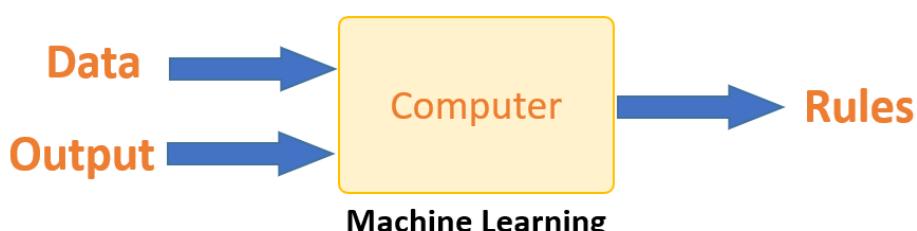
## บทนำ

### 1.1 ความเป็นมาและความสำคัญ

ในทางวิทยาการคอมพิวเตอร์ ศาสตร์ทางด้านการเรียนรู้ของเครื่องจักร (Machine Learning) จัดเป็นส่วนหนึ่งของปัญญาประดิษฐ์ที่นักวิทยาการคอมพิวเตอร์ต้องการสร้างขั้นตอนวิธีการเรียนรู้จากข้อมูลในอดีต ด้วยตัวแบบ (Model) ที่อาจอยู่ในรูปของสมการโครงสร้างต้นไม้ การเชื่อมโยงในเครือข่าย แล้วนำไปประยุกต์กับข้อมูลที่ยังไม่พบในอนาคต โดยความแตกต่างระหว่างการพัฒนาโปรแกรมเพื่อแก้ปัญหาการคำนวณกับการพัฒนาโปรแกรมที่เรียนรู้ได้ด้วยเครื่องจักรคือ การเขียนโปรแกรมเพื่อแก้ปัญหาการคำนวณ มีการกำหนดกฎกติกาและขั้นตอนการประมวลผลเพื่อให้ได้ผลลัพธ์ที่ต้องการ เมื่อมีข้อมูลที่ป้อนเข้ามาตรงกับกฎที่สร้างไว้โปรแกรมจะทำการประมวลผลแล้วให้ผลลัพธ์ตามที่กำหนดแต่แรกดูภาพที่ 1 ถ้ามีการเพิ่มระบบที่ซับซ้อนขึ้นก็มีการพัฒนาเพิ่มสำหรับการเรียนรู้ด้วยเครื่องจักรจะไม่มีการกำหนดกฎกติกาและรูปแบบการคำนวณที่ชัดเจนไว้ แต่จะมีการใช้ตัวแบบที่มีพารามิเตอร์ซึ่งจะถูกเปลี่ยนตามข้อมูลในอดีตทำให้เกิดการสร้างกฎใหม่ขึ้นมาเองได้ตามภาพที่ 2 ผู้พัฒนาโปรแกรมไม่จำเป็นต้องเขียนกฎใหม่ทุกครั้งที่มีข้อมูลใหม่เพิ่มมา ขั้นตอนวิธีการเรียนรู้จะปรับพารามิเตอร์ให้เข้ากับข้อมูลใหม่เองอัตโนมัติ[1]



ภาพที่ 1.1 รูปแบบการทำงานโปรแกรมแบบดั้งเดิม



ภาพที่ 1.2 รูปแบบการทำงานการเรียนรู้ด้วยเครื่องจักร

การเรียนรู้ด้วยเครื่องจักร แบ่งออกเป็นได้ 3 ประเภทได้แก่

1. การเรียนรู้แบบมีผู้สอน (Supervised learning) จะใช้ข้อมูลในการปรับพารามิเตอร์เริ่มต้น (training data) ตัวแบบจะคำนวณค่าของตัวอย่าง แล้วนำมาปรับปรุงพารามิเตอร์ให้เหมาะสม การเรียนรู้แบบมีผู้สอนแยกออกเป็น 2 ประเภทย่อย

- การจำแนกประเภท (Classification) ตัวแปรเป้าหมายมีค่าไม่ต่อเนื่อง โดยตัวจำแนกประเภท (classifier) จะคำนวณค่าตัวแปรเป้าหมาย
- สมการการถดถอย (Regression) โดยที่ตัวแปรเป้าหมายมีค่าต่อเนื่อง ยกตัวอย่างเช่น การคำนวณค่าของหุ้น

ตัวอย่างตัวแบบของการเรียนรู้แบบมีผู้สอน เช่น สมการถดถอยเชิงเส้น (Linear Regression), สมการถดถอยโลจิสติกส์ (Logistic Regression), ต้นไม้การตัดสินใจ (Decision Tree) และป่าสุ่ม (Random Forest) เครือข่ายประสาท (Neural network) เป็นต้น

2. การเรียนรู้แบบไม่มีผู้สอน (Unsupervised learning) จะไม่มีการกำหนดตัวแปรเป้าหมาย การเรียนรู้ที่เกิดขึ้นเป็นการจัดกลุ่มของข้อมูลแยกเป็นแต่ละส่วน เช่น การแยกข้อมูลประชากรออกเป็นส่วน ๆ ที่สัมพันธ์กัน การใช้งานหลักมี 2 อย่างคือ

- การลดมิติของข้อมูล (Dimensionality reduction) เน้นการลดความซับซ้อนของปริมาณข้อมูลก่อนนำไปใช้ เพื่อนำมาแสดงในกราฟที่มีมิติที่ต่ำลงได้
- การวิเคราะห์การเกาะกลุ่ม (Clustering analysis) เน้นการรวมข้อมูลเป็นกลุ่ม ตามลักษณะต่าง ๆ ของตัวอย่าง เช่น การจัดกลุ่มลูกค้าตามพฤติกรรมการซื้อ

ตัวอย่างของการเรียนรู้แบบไม่มีผู้สอน เช่น การเกาะกลุ่มค่าเฉลี่ยเค (K-means clustering), ตัวแบบผสม gauss (Gaussian mixture model), การเกาะกลุ่มตามลำดับชั้น (Hierarchical cluster) และ การวิเคราะห์แคนนอนสำคัญ (Principal component analysis) เป็นต้น

3. การเรียนรู้แบบเสริมกำลัง (Reinforcement learning) เน้นการเรียนรู้วิธีการตัดสินใจภายใต้สถานการณ์ต่าง ๆ โดยมีเป้าหมายสุดท้ายคือรางวัลที่มากจากผลของการกระทำในสถานการณ์ต่างๆ ทำให้เกิดการค้นหาแนวทางรับมือกับปัญหาที่ดีที่สุด ตัวอย่างการใช้งาน เช่น การเล่นโกะของ AlphaGO, รถที่ขับเคลื่อนได้ด้วยตนเอง (self-driving car) และบอทแลกเปลี่ยนสต็อก (stock trading bot) เป็นต้น

ในทุกๆ รูปแบบของการเรียนรู้ของเครื่อง เป้าหมายหลักคือการสร้างโมเดลที่สามารถทำงานผลลัพธ์หรือตัดสินใจอย่างมีประสิทธิภาพ โดยพิจารณาจากคุณลักษณะ (features) ของข้อมูลที่มีอยู่

การเรียนรู้ของเครื่องเป็นเทคโนโลยีที่มีผลกระทบมากในวงการวิทยาศาสตร์และเทคโนโลยี โดยเฉพาะในสาขางานที่ต้องการการตัดสินใจอัตโนมัติ หรือการทำนายผลลัพธ์จากข้อมูลใหญ่ที่ซับซ้อน ตัวอย่างเช่น การวิเคราะห์ข้อมูลการเคลื่อนไหวของหุ่น การทำนายการซื้อขายของลูกค้า หรือการทำนายการแพร่กระจายของโรคระบาด

วิธีการเกาะกลุ่มหรือการจัดกลุ่ม (Clustering) เป็นหนึ่งในเทคนิคพื้นฐานในการเรียนรู้ของเครื่องแบบไม่มีผู้สอน (Unsupervised Learning) ที่ใช้สำหรับการค้นหาโครงสร้างที่ซ่อนอยู่ในชุดข้อมูลที่ซับซ้อนและไม่มีป้ายกำกับ ซึ่งจุดประสงค์หลักของการจัดกลุ่มคือการจัดระเบียบข้อมูลที่มีลักษณะคล้ายคลึงกันให้อยู่ในกลุ่มเดียวกัน

วิธีการเกาะกลุ่มมีการใช้งานที่หลากหลาย ทั้งในด้านวิทยาศาสตร์ วิศวกรรม ธุรกิจ และสาขางานอื่น ๆ ด้วยความสามารถในการจัดการกับข้อมูลที่มีมิติมาก และสามารถหาความสัมพันธ์ที่ซับซ้อนระหว่างข้อมูล การเกาะกลุ่มถูกใช้ในการวิเคราะห์ข้อมูลทางสังคม การจัดการฐานข้อมูล การจำแนกประเภทข้อมูลภาพและวิดีโอ การวิเคราะห์ข้อมูลชีวภาพ และอื่น ๆ

อย่างไรก็ตาม ตัวเลือกของวิธีการเกาะกลุ่มที่เหมาะสมจะขึ้นอยู่กับลักษณะของข้อมูลและความต้องการของการใช้งาน วิธีการเกาะกลุ่มที่หลากหลายอาจมีความซับซ้อนและความคล้ายคลึงที่แตกต่างกัน ซึ่งทำให้การเลือกวิธีการที่เหมาะสมสมำรถรับการจัดกลุ่มข้อมูลเป็นเรื่องที่ท้าทาย

สำหรับโครงการนี้เลือกการใช้ MOF มากำหนดค่าความผิดปกติของข้อมูลแล้วนำข้อมูลมาหารูปแบบเกาะกลุ่มด้วยวิธีการต่างๆ ซึ่งเป็นการเรียนรู้แบบไม่มีผู้สอน โดยในการหาค่าผิดปกติของข้อมูลนั้นจะกำหนดค่าเบอร์เซ็นไทล์เพื่อใช้เป็นเกณฑ์ในการกำหนดจุดผิดปกติ จุดที่มีค่า MOF score ที่สูงก็มีแนวโน้มที่จะเป็นจุดผิดปกติสูงมาก

โครงการนี้นำเสนอขั้นตอนวิธีการตรวจจับข้อมูลผิดปกติก่อนส่งไปประมวลผลขั้นตอนวิธีการเกาะกลุ่มซึ่งพบว่า ขั้นตอนวิธีการเกาะกลุ่มให้ค่าซิลูอุทที่ประเมินการเกาะกลุ่มที่มีประสิทธิภาพกว่ากับข้อมูลสังเคราะห์ที่มีข้อมูลผิดปกติปะปนอยู่แต่ไม่ได้ประสิทธิภาพที่ดีกับข้อมูล Spotify ที่มีการเกาะกับข้อมูลที่หนาแน่น

ในโครงการนี้จะทำการทดสอบการทำงานร่วมกันระหว่างขั้นตอนวิธีการตรวจจับข้อมูลผิดปกติ MOF กับขั้นตอนวิธีการเกาะกลุ่มเคมีนและขั้นตอนวิธีการเกาะกลุ่มแบบลำดับขั้น โดยเขียนโปรแกรมการทดสอบผ่านทางภาษา Python ทดสอบกับข้อมูลสังเคราะห์ทั้งรูปแบบ (1) วงกลมช้อนสองวง (2) รูปพระจันทร์ช้อนสองวง (3) กลุ่มข้อมูลกระจายสามกลุ่ม (4) กลุ่มข้อมูลที่เรียงเป็นเส้นสี่เส้น (5) กลุ่มข้อมูลที่กระจายเป็นกลุ่มใหญ่ และทดสอบกับข้อมูลโลจิริงจาก Kaggle ขนาดข้อมูล 42,305 ตัว

## บทที่ 2

### แนวคิด ทฤษฎี และงานวิจัยที่เกี่ยวข้อง

#### 2.1 แนวคิดและทฤษฎีที่เกี่ยวข้อง

##### 2.1.1 การวิเคราะห์การเกาะกลุ่ม

การเกาะกลุ่มเป็นเทคนิคแบบหนึ่งของการเรียนรู้แบบไม่มีผู้สอน โดยการเกาะกลุ่มนั้นจะจัดกลุ่มข้อมูลที่มีความคล้ายคลึงกันให้อยู่ในกลุ่มเดียวกัน วัตถุประสงค์การแบ่งกลุ่มข้อมูลนั้นคือ การหารูปแบบหรือโครงสร้างบางอย่างที่อยู่ภายในข้อมูล ตามความสัมพันธ์หรือจัดประเภทของข้อมูลออกมา การเกาะกลุ่มมักจะถูกใช้ในการวิเคราะห์ข้อมูล การหารูปแบบพร้อมการตรวจจับข้อมูลที่ผิดปกติ [2]

ขั้นตอนวิธีการเกาะกลุ่มนั้นมีหลายวิธีโดยวิธีที่เป็นที่นิยมได้แก่

1) ขั้นตอนวิธีเคมีน : ขั้นตอนวิธีการเริ่มจากการกำหนดจำนวนจุดศูนย์กลางและทำการระบะทางที่ใกล้ที่สุดระหว่างจุดข้อมูลกับจุดศูนย์กลางกลุ่มที่ตั้งขึ้นมา โดยจุดข้อมูลใดๆที่อยู่ไกลจุดศูนย์กลางกลุ่มไปหนักจะถูกตั้งให้จุดข้อมูลนั้นอยู่ในกลุ่มของจุดศูนย์กลางนั้น และต่อมาทำการย้ายจุดศูนย์กลางกลุ่มด้วยค่าเฉลี่ยภายในกลุ่มของตนเอง ซึ่งขั้นตอนเหล่านี้จะถูกทำซ้ำเรื่อยๆเพื่อหาค่า the within-cluster sum of squares (WCSS) น้อยที่สุด ซึ่งขั้นตอนวิธีเคมีน นั้นจะมีความอ่อนไหวอย่างมากต่อข้อมูลที่ผิดปกติ (outlier) เนื่องจากทำให้เกิดการกำหนดจุดศูนย์กลางกลุ่มที่เปลกไปจากความเป็นจริง เพราะข้อมูลที่ผิดปกติจะตึงค่าจุดศูนย์กลางออกส่างผลให้การจัดกลุ่มที่เกิดขึ้นนั้นแย่ลง และการหา the within-cluster sum of squares ถูกผลกระทบจากตัวข้อมูลผิดปกติอาจส่งผลให้เกิดการจัดกลุ่มที่ไม่ดีขึ้น [2]

2) ขั้นตอนวิธีการเกาะกลุ่มระดับชั้น : ขั้นตอนวิธีนี้จะเกาะกลุ่มข้อมูลตามกับโครงสร้างต้นไม้เป็นประเภทหนึ่งของการเรียนรู้โดยไม่มีผู้สอนโดยจะแบ่งกลุ่มตามความคล้ายคลึงกันของข้อมูล โดยดูจากการวัดระยะระหว่างจุดข้อมูลหรือระหว่างกลุ่ม ซึ่งวิธีการวัดระยะนั้นมีหลายรูปแบบโดยทั่วไปนั้นจะใช้วิธีการวัดระยะทางแบบยุคลิด เป็นการวัดระยะห่างระหว่างสองจุดในปริภูมิ  $n$  มิติ เราสามารถคำนวณระยะทางแบบยุคลิดระหว่างจุด A และ B ได้โดยใช้สูตรต่อไปนี้:

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}$$

โดยที่  $q$  และ  $p$  คือสองจุดที่ต้องการหาระยะทาง ,  $(p_1, p_2, \dots, p_n)$  เป็นพิกัดของจุด  $p$  และ  $(q_1, q_2, \dots, q_n)$  เป็นพิกัดของจุด  $q$

ขั้นตอนวิธีการแกะกลุ่มระดับชั้น มีวิธีการหาความคล้ายคลึงกันของข้อมูลตามลำดับชั้นด้วยกัน 2 วิธีหลัก ได้แก่

การแกะกลุ่มแบบต่อขึ้น (Agglomerative Clustering): เริ่มต้นด้วยจุดข้อมูลแต่ละจุดเป็นกลุ่มของตัวเอง แล้วทำการรวมกลุ่มที่ใกล้กันที่สุดเข้าด้วยกัน สิ่งนี้จะทำซ้ำๆ จนกระทั่งทุกกลุ่มรวมเข้าด้วยกันเป็นกลุ่มเดียว การคำนวณความใกล้กันของกลุ่มสามารถใช้เกณฑ์ต่างๆ เช่น ระยะทางแบบยุคลิด, ความคล้ายคลึงความสัมพันธ์, และ ระยะทางมหาราโนบิส

"Linkage" หรือ การเชื่อมโยงในบริบทของการแกะกลุ่มระดับชั้น เป็นกระบวนการที่ใช้ในการรวมกลุ่มข้อมูล โดยมีหลักการคำนวณความใกล้กันหรือความคล้ายคลึงระหว่างกลุ่มใหม่ที่กำลังจะรวมกัน มีหลายวิธีการเชื่อมโยงที่นิยมใช้ในการแกะกลุ่มระดับชั้น

โดยกำหนดให้

- $d(A, B)$  คือ การแกะกลุ่มกันระหว่างกลุ่ม  $A$  และ  $B$
- $a$  คือ จุดข้อมูลภายในกลุ่ม  $A$
- $b$  คือ จุดข้อมูลภายในกลุ่ม  $B$
- $dist(a, b)$  คือ ระยะทางระหว่างจุดข้อมูล  $a$  กับ  $b$  เช่น ระยะทางแบบยุคลิด

Single Linkage (ระยะทางต่ำสุด): คำนวณระยะทางระหว่างจุดที่ใกล้กันที่สุดในแต่ละกลุ่ม แล้วนำระยะทางที่น้อยที่สุดมาเป็นเกณฑ์ในการเชื่อมโยงกลุ่มก็คือ

$$d(A, B) = \min_{a \in A, b \in B} \{dist(a, b)\}$$

Complete Linkage (ระยะทางสูงสุด): คำนวณระยะทางระหว่างจุดที่ห่างกันที่สุดในแต่ละกลุ่ม แล้วนำระยะทางที่มากที่สุดมาเป็นเกณฑ์ในการเชื่อมโยงกลุ่มก็คือ

$$d(A, B) = \max_{a \in A, b \in B} \{dist(a, b)\}$$

Average Linkage (ระยะทางเฉลี่ย): คำนวณระยะทางเฉลี่ยระหว่างจุดในแต่ละกลุ่ม แล้วนำระยะทางเฉลี่ยนี้มาเป็นเกณฑ์ในการเชื่อมโยงกลุ่มก็คือ

$$d(A, B) = \frac{1}{|A||B|} \sum_{\substack{a \in A \\ b \in B}} dist(a, b)$$

Centroid Linkage (ระยะทางเซนทรอยด์): เป็นการคำนวณระยะทางระหว่างจุดศูนย์กลางกลุ่ม โดยใช้ระยะทางระหว่างจุดศูนย์กลางกลุ่มเป็นเกณฑ์ในการเชื่อมโยงกลุ่ม

$$d(A, B) = dist(centroid(A), centroid(B))$$

การเลือกวิธีเชื่อมโยง (linkage) ในการเกาะกลุ่มระดับชั้น ขึ้นอยู่กับลักษณะข้อมูลและความต้องการของผู้วิเคราะห์ โดยสิ่งสำคัญคือการเลือกวิธีเชื่อมโยงที่เหมาะสมจะทำให้ผลลัพธ์ของการวิเคราะห์มีความแม่นยำและเป็นไปตามความต้องการของผู้ใช้งานมากขึ้น ซึ่ง Agglomerative single linkage ก็คืออัลกอริทึมรูปแบบหนึ่งของขั้นตอนวิธีการเกาะกลุ่มระดับชั้น ซึ่งสามารถจัดกลุ่มข้อมูลที่มีรูปร่างเปลกๆ ออกมайд้วยกันได้แต่อาจเกิดปัญหาเมื่อข้อมูลเหล่านั้นมีข้อมูลที่ผิดปกติ (outlier) อยู่ก่อให้เกิดปัญหา chaining effects ขึ้นมา ดังนั้น Agglomerative single linkage จึงมีความอ่อนไหวต่อข้อมูลที่ผิดปกติอย่างมาก

### 2.1.2 การเตรียมการข้อมูล

ก่อนที่จะนำข้อมูลมาวิเคราะห์การเกาะกลุ่มนั้นมักจะต้องมีการจัดการกับข้อมูลก่อน เพื่อที่จะสามารถจัดกลุ่มข้อมูลได้อย่างมีประสิทธิภาพ ซึ่งก็มีหลากหลายวิธีการในการเตรียมข้อมูล

1) การทำข้อมูลให้สะอาด: ข้อมูลไม่ปกติ เช่น การไม่ปราศจากข้อมูล ข้อมูลที่มีค่ามากหรือน้อยเกินไป ข้อมูลที่ซ้ำซ้อน

2) การสเกลข้อมูล: หมายถึงการปรับขนาดข้อมูลให้อยู่ในช่วงที่กำหนด หรือให้มีความสอดคล้องกันเพื่อควบคุมความผันผวน การสเกลข้อมูลมีความสำคัญในการวิเคราะห์ข้อมูลหลายมิติ เนื่องจากความแตกต่างของหน่วยวัดและช่วงค่าของแต่ละมิติอาจส่งผลกระทบต่อผลลัพธ์ของการวิเคราะห์ มีวิธีการสเกลข้อมูลที่นิยมใช้งานได้แก่[3]

1. Min-Max Scaling: ปรับข้อมูลให้อยู่ในช่วงที่กำหนดโดยทั่วไปแล้วจะให้อยู่ในช่วง 0 ถึง 1 ใช้สูตรดังต่อไปนี้

$$X' = (X - \min(X)) / (\max(X) - \min(X))$$

โดยที่  $X$  คือข้อมูลตั้งต้น และ  $X'$  คือข้อมูลที่ถูกสเกล

2. Standard Scaling (Z-score Scaling): ปรับข้อมูลให้มีค่าเฉลี่ยเป็น 0 และค่าความแปรปรวนเป็น 1 โดยใช้สูตร

$$X' = (X - \text{mean}(X)) / \text{std}(X)$$

โดยที่  $X$  คือข้อมูลตั้งต้น และ  $X'$  คือข้อมูลที่ถูกสเกล

3. Max Abs Scaling: ปรับข้อมูลโดยหารด้วยค่าสูงสุดของข้อมูล ให้ค่าใหม่อยู่ในช่วง -1 ถึง 1 โดยใช้สูตร

$$X' = X / \max(\text{abs}(X))$$

โดยที่  $X$  คือข้อมูลตั้งต้น และ  $X'$  คือข้อมูลที่ถูกสเกล

4. Robust Scaling: ปรับข้อมูลโดยลดผลกระทบของข้อมูลผิดปกติ (outliers) โดยใช้ค่าวอไทร์ล์ (quartiles) ในการสเกลข้อมูล สูตรที่ใช้คือ

$$X' = (X - Q1(X)) / (Q3(X) - Q1(X))$$

โดยที่  $X$  คือข้อมูลตั้งต้น,  $X'$  คือข้อมูลที่ถูกสเกล,  $Q1(X)$  คือค่าวอไทร์ล์ที่ 1 (25th percentile) ของข้อมูล, และ  $Q3(X)$  คือค่าวอไทร์ล์ที่ 3 (75th percentile) ของข้อมูล

การเลือกวิธีการสเกลข้อมูลขึ้นอยู่กับลักษณะของข้อมูลและความต้องการของการวิเคราะห์ การสเกลข้อมูลอาจช่วยให้ผลลัพธ์ของการวิเคราะห์มีความแม่นยำมากขึ้น และเพิ่มประสิทธิภาพในกระบวนการวิเคราะห์ เช่น การวิเคราะห์เชิงเส้น, การเกากลุ่ม, หรือการวิเคราะห์องค์ประกอบหลัก (PCA)

4) การกำหนดค่าพารามิเตอร์และการเลือกใช้ขั้นตอนวิธีการเกากลุ่มให้เหมาะสม: การเลือกขั้นตอนวิธีการเกากลุ่มให้เหมาะสมกับข้อมูลนั้นเป็นสิ่งจำเป็นต่อการจัดกลุ่มข้อมูลโดยการเลือกนั้นขึ้นอยู่กับปัญหา และความต้องการของปัญหา การประมาณค่าจำนวนกลุ่มที่ใช้ในการเกากลุ่มด้วยวิธีการ K-mean ก็เป็นหนึ่งในการเลือกพารามิเตอร์ วิธีต่างๆ ที่ช่วยในการเลือกตัวอย่างเช่น elbow method, silhouette analysis หรือ gap statistics เป็นต้น[4]

### 2.1.3 การประเมินผลการเกากลุ่ม

การประเมินผลการเกากลุ่มข้อมูลเป็นอีกขั้นตอนหนึ่งที่สำคัญในการจัดกลุ่มข้อมูลต่างๆ การประเมินผลการจัดกลุ่มนั้นนิ่งไว้เพื่อคุณภาพของกลุ่มข้อมูลที่จัดออกมานี้และคุณประสิทธิภาพของขั้นตอนวิธีการเกากลุ่ม ซึ่งมีหลากหลายวิธีในการประเมินผลการเกากลุ่มข้อมูล โดยวิธีการหลักๆ มีดังต่อไปนี้

Silhouette score: เป็นการวัดความใกล้ชิดกันภายในกลุ่มข้อมูลและวัดความห่างไกลกันของข้อมูลที่อยู่ต่างกลุ่มกัน โดยค่า Silhouette score คำนวณจากการหาค่าเฉลี่ยของระยะทางแต่ละจุดภายในกลุ่มเรียกว่า  $a(i)$  และหาค่าเฉลี่ยที่ต่ำที่สุดระหว่างจุดข้อมูลระหว่างกลุ่มเรียกว่า  $b(i)$  นำ  $a(i)$  ลบ  $b(i)$  แล้วหารด้วยค่าสูงสุดระหว่าง  $a(i)$  กับ  $b(i)$  เราก็จะได้ค่า Silhouette ออกมาก โดยค่า Silhouette จะอยู่ในช่วง -1 ถึง 1 ถ้าค่า Silhouette มีค่าใกล้ 1 ก็หมายความว่าเป็นการจัดกลุ่มที่ดี ในขณะที่ถ้าค่า Silhouette มีค่าใกล้ -1 ก็จะบ่งบอกได้ว่าข้อมูลที่ถูกจับกลุ่มกันนั้นอยู่กันผิดกลุ่ม และค่า Silhouette เท่ากับ 0 จะสื่อถึงการไม่มีขอบเขตซึ่งกันและกันระหว่างกลุ่ม[8] โดยสูตรการคำนวณ Silhouette score สำหรับจุดข้อมูลเดียว (silhouette coefficient) คือ:

$$s(i) = (b(i) - a(i)) / \max(a(i), b(i))$$

โดยที่:

$i$  คือจุดข้อมูล

$a(i)$  คือค่าเฉลี่ยระยะทางระหว่างจุดข้อมูล  $i$  และจุดข้อมูลอื่น ๆ ในกลุ่มเดียวกัน (ความคลุมเคลือภายในกลุ่ม)

$b(i)$  คือค่าเฉลี่ยระยะทางระหว่างจุดข้อมูล  $i$  และจุดข้อมูลในกลุ่มที่ใกล้ที่สุด (ความคลุมเคลือระหว่างกลุ่ม)

สำหรับการคำนวณ Silhouette score สำหรับทุก ๆ จุดข้อมูลในชุดข้อมูล ให้ทำตามขั้นตอนต่อไปนี้:

1. คำนวณค่า silhouette coefficient ( $s(i)$ ) สำหรับแต่ละจุดข้อมูล  $i$  ในชุดข้อมูล โดยใช้สูตรด้านบน
2. คำนวณค่าเฉลี่ยของค่า silhouette coefficient ที่คำนวณได้ในขั้นตอนที่ 1 สำหรับทุก ๆ จุดข้อมูล

Silhouette score คือค่าเฉลี่ยของค่า silhouette coefficient ของทุก ๆ จุดข้อมูลในชุดข้อมูล

#### 2.1.4 ข้อมูลสังเคราะห์จาก scikit-learn และข้อมูล spotify

Scikit-learn เป็นไลบรารีที่ใช้สำหรับการเรียนรู้ของเครื่อง และวิเคราะห์ข้อมูล (data analysis) บนภาษา Python มีคุณสมบัติในการสนับสนุนการสร้างข้อมูลสังเคราะห์ (synthetic data) ซึ่งเป็นข้อมูลที่สร้างขึ้นโดยอาศัยความสัมพันธ์ที่กำหนดให้ ข้อมูลนี้สามารถนำไปใช้ประโยชน์ในงานวิจัยหรือการพัฒนาโมเดลทางด้านวิทยาศาสตร์ข้อมูล [6]

การทดสอบขั้นตอนวิธีการแก้กลุ่มข้อมูลนอกจากจะทดสอบกับข้อมูลสังเคราะห์แล้วเรายังต้องทดสอบกับข้อมูลในโลกแห่งความเป็นจริงเพื่อดูการทำงานของการจัดกลุ่มและเพื่อยกรุ้งการแบ่งกลุ่มในข้อมูล โดยข้อมูล

ที่มักจะถูกนำมาใช้ในการจัดกลุ่มข้อมูลคือ ข้อมูล Spotify เป็นข้อมูลที่ได้มาจากการสตรีมเพลงออนไลน์ของ Spotify ซึ่งประกอบด้วยข้อมูลเชิงปริมาณและเชิงคุณภาพของเพลง ซึ่งลักษณะข้อมูลใน Spotify [7] มีดังต่อไปนี้

1) danceability: ค่าที่วัดความเหมาะสมของเพลงสำหรับการเต้น ค่าสูงหมายความว่าเพลงนั้นเหมาะสมสำหรับการเต้นมากมีค่าอยู่ระหว่าง 0 ถึง 1

2) energy: ค่าที่วัดความกระตือรือร้นและความแรงของเพลง ค่าสูงหมายความว่าเพลงมีพลังงานมากมีค่าอยู่ระหว่าง 0 ถึง 1

3) key: คีย์หรือโน้ตพื้นฐานของเพลง

4) loudness : ค่าที่วัดความดังของเพลง วัดเป็นเดซิเบล (dB) มีค่าอยู่ระหว่าง -60 ถึง 0 dB

5) mode : ค่าที่ระบุความสัมพันธ์ระหว่างคีย์และสเกล เช่น สเกลเมเจอร์ (1) หรือสเกลไมเนอร์ (0)

6) speechiness : ค่าที่วัดการพูดของเสียง ค่าสูงหมายความว่ามีการพูดที่เยอะมากกว่าเสียงเพลงมีค่าอยู่ระหว่าง 0 ถึง 1

7) acousticness : เป็นค่าที่วัดเสียงอิเล็กทรอนิกส์โดยขึ้นอยู่เสียงอิเล็กทรอนิกส์โดยค่าที่สูงแสดงถึงการมีเสียงอิเล็กทรอนิกส์ที่เยอะมีค่าอยู่ระหว่าง 0 ถึง 1

8) instrumentalness : ค่าที่วัดความน่าจะเป็นของเพลงที่ไม่มีการร้องเพลง ค่าสูงหมายความว่าเพลงนั้นมีเสียงเครื่องดนตรีมากและมีเสียงร้องของมนุษย์น้อยมีค่าอยู่ระหว่าง 0 ถึง 1

9) liveness : ค่าที่วัดความมีชีวิตชีวาของเพลง ค่าสูงหมายความว่าเพลงนั้นมีความมีชีวิตชี瓦มากมีค่าอยู่ระหว่าง 0 ถึง 1

10) valence : ค่าวัดเพลงที่มีแนวโน้มไปด้านบวกหรือด้านการให้ความสุข โดยพื้นฐานมาจากทำงานของเพลง เนื้อเพลง และจังหวะของเพลงมีค่าอยู่ระหว่าง 0 ถึง 1 ค่าสูงแสดงถึงเพลงที่มีการแสดงอารมณ์ทางบวกและให้ความสุขที่สูง

11) tempo : ความเร็วของเพลง วัดเป็นจังหวะต่อนาที (BPM)

12) type, id, uri, track\_href, analysis\_url: เป็นข้อมูลเฉพาะที่เชื่อมโยงกับแทร็คเพลงใน Spotify

13) duration\_ms : ความยาวของเพลง วัดเป็นมิลลิวินาที

14) time\_signature : จำนวนจังหวะในแต่ละช่วงของเพลง

15) genre (แนวเพลง): ประเภทหรือแนวเพลง เช่น ป็อป ร็อก แจ๊ส ฯลฯ

16) song\_name, Unnamed: 0, title: ข้อมูลเกี่ยวกับชื่อเพลง อาจมีการเก็บเข้ากัน

ในโครงการนี้นำข้อมูลสังเคราะห์จาก Scikit-learn และข้อมูล Spotify มาใช้ประโยชน์ในด้านต่าง ๆ ดังนี้

1. การวิเคราะห์ข้อมูล: ค้นหาความสัมพันธ์ในข้อมูล Spotify เพื่อหาแนวโน้มหรือความสัมพันธ์ที่น่าสนใจ เช่น ความนิยมของเพลงกับค่าความน่าสนใจของศิลปิน

2. การสร้างโมเดลการเรียนรู้ของเครื่อง: ใช้ข้อมูล Spotify ในการสร้างโมเดลเพื่อทำนายความนิยมของเพลงหรือแนวนำเสนอเพลงให้กับผู้ใช้งานตามความชอบ และสามารถใช้ข้อมูลสังเคราะห์จาก Scikit-learn ในการทดสอบและปรับปรุงโมเดลเพื่อเพิ่มประสิทธิภาพและความแม่นยำของโมเดล

3. การประเมินผลของโมเดล: นำข้อมูลสังเคราะห์จาก Scikit-learn มาใช้ในการประเมินผลโมเดลที่สร้างขึ้นจากข้อมูล Spotify เพื่อให้แน่ใจว่าโมเดลที่สร้างขึ้นสามารถทำงานได้ดีกับข้อมูลที่ไม่เคยพบมาก่อน

4. การค้นหาแนวโน้มใหม่: นำข้อมูลสังเคราะห์จาก Scikit-learn มาเปรียบเทียบกับข้อมูล Spotify เพื่อค้นหาแนวโน้มใหม่ ๆ ที่อาจมีผลต่อความนิยมของเพลง และสามารถนำไปปรับปรุงโมเดลหรือนำไปใช้ในงานวิจัยต่อไป

5. การปรับปรุงข้อมูล: ใช้ข้อมูลสังเคราะห์จาก Scikit-learn เป็นแหล่งข้อมูลเสริมเพื่อเพิ่มปริมาณข้อมูลในการวิเคราะห์ข้อมูล Spotify ทำให้การวิเคราะห์มีความหลากหลายและครอบคลุมมากขึ้น

โดยรวมแล้ว การนำข้อมูลสังเคราะห์จาก Scikit-learn และข้อมูล Spotify มาใช้ในโครงการสามารถช่วยให้สามารถสร้างโมเดลที่มีประสิทธิภาพและความแม่นยำสูงขึ้น นอกจากนี้ยังช่วยให้สามารถค้นคว้าความสัมพันธ์และแนวโน้มที่น่าสนใจในข้อมูลเพลงของ Spotify และเข้าใจลักษณะข้อมูลที่ส่งผลต่อความนิยมของเพลง หรือการแนะนำเพลงให้กับผู้ใช้งานได้ดียิ่งขึ้น

### 2.1.5 การสุ่มตัวอย่างข้อมูล (Random sampling)

การสุ่มตัวอย่างข้อมูลนั้นเป็นเทคนิคที่ใช้กันในทางสถิติและทางด้านการวิเคราะห์ข้อมูล เพื่อจะหยิบข้อมูลจากประชากรหรือชุดข้อมูลที่มีขนาดใหญ่ หลักการสุ่มตัวอย่างข้อมูลคือการสุ่มหยิบข้อมูลโดยที่ข้อมูลทุกตัวนั้นมีโอกาสสูงที่จะถูกเลือก เท่าๆ กัน การทำแบบนี้เพื่อที่จะให้แน่ใจได้ว่าข้อมูลที่หยิบออกมานั้นจะสามารถแทนข้อมูลทั้งหมดได้

และเพื่อลดค่าใช้จ่ายในการใช้ข้อมูลนั้นๆ การสุ่มตัวอย่างข้อมูลนั้นเมื่อถูกกันหลาบวิธีเช่น simple random sampling, stratified random sampling และ cluster sampling เป็นต้น ในกรณีที่มีขนาดไม่ใหญ่มาก Systematic sampling ก็เป็นอีกวิธีการหนึ่งที่เหมาะสมโดยหลักการการสุ่มของวิธีนี้ก็คือ การหยิบเว้นช่วงระยะเท่าๆกัน เทคนิคนี้เริ่มจากการสุ่มเลือกจุดเริ่มต้นที่จะสุ่มข้อมูล แล้วกำหนดค่าจำนวนข้อมูลทั้งหมดที่เราต้องการอุปมา นำจำนวนข้อมูลทั้งหมดที่เราต้องการมาหารกับจำนวนข้อมูลที่ต้องการ เราก็จะได้ลำดับขั้นในการหยิบข้อมูลอุปมา[8]

ข้อดี:

- 1) เป็นวิธีการที่เข้าใจง่ายและมีประสิทธิภาพโดยเฉพาะอย่างยิ่งการใช้เทคนิคนี้กับข้อมูลที่มีการเรียงลำดับ
- 2) การสุ่มหยิบนี้มีแนวโน้มที่จะสร้างอุปต่อข้อมูลที่น้อย วิธีการนี้หลีกเลี่ยงการเลือกข้อมูลเดิมโดยบังเอิญ
- 3) การสุ่มหยิบนี้สามารถบ่งบอกได้ถึงข้อมูลจริงได้เป็นอย่างดี

ข้อเสีย:

- 1) ถ้าข้อมูลมีรูปแบบบางอย่างที่ซ่อนอยู่อาจทำให้เกิดผลลัพธ์ที่มีอุปต์ได้
- 2) การสุ่มด้วยวิธีนี้จำเป็นต้องมีการกำหนดลำดับของข้อมูลที่เหมาะสมก่อน

### 2.1.6 เปอร์เซ็นไทล์ (percentile)

เปอร์เซ็นไทล์นั้นเป็นแนวคิดทางด้านสถิติมักจะใช้ในการอธิบายถึงความสัมพันธ์บางอย่างของมาจากชุดข้อมูล เปอร์เซ็นไทล์นั้นเป็นการบอกถึงเปอร์เซ็นของจุดข้อมูลที่มีค่าสูงกว่าหรือต่ำกว่าค่าเฉลี่ยบางค่า ตัวอย่างเช่น เปอร์เซ็นไทล์ที่ 25 หรือเรียกอีกอย่างว่าควอไทล์แรก (first quartile) บ่งบอกถึงค่า 25 % ของจุดข้อมูลใดๆที่มีค่าต่ำกว่าหรือเท่ากับค่านั้น

เปอร์เซ็นไทล์มักถูกใช้ในการสรุปและอธิบายการแจกแจงของข้อมูล ระบุแนวโน้ม และตรวจหาข้อมูลที่ผิดปกติ นอกจากนี้เปอร์เซ็นไทล์ยังเป็นเครื่องมือที่มีประโยชน์เมื่อข้อมูลที่ไม่ได้แจกแจงตามรูปแบบปกติ เนื่องจากเปอร์เซ็นไทล์ไม่ขึ้นอยู่กับข้อสมมติเกี่ยวกับการแจกแจงที่ซ่อนอยู่[9]

วิธีการคำนวณค่าเปอร์เซ็นไทล์ที่  $p$  มีขั้นตอนดังนี้:

- 1) เรียงลำดับข้อมูลจากน้อยไปมาก
- 2) คำนวณค่าดัชนี (index) สำหรับค่าเปอร์เซ็นไทล์ที่  $p$  โดยใช้สูตร  $\text{index} = (p/100) * (n + 1)$  โดยที่  $p$  คือเปอร์เซ็นต์ที่ต้องการหา (เช่น 25, 50, 75) และ  $n$  คือจำนวนข้อมูลทั้งหมดในชุดข้อมูล

### 3) ตรวจสอบค่าดัชนีที่คำนวณได้

- ถ้าค่าดัชนีเป็นจำนวนเต็ม ให้ค่าเบอร์เซ็นไทล์ที่  $p$  เท่ากับค่าข้อมูลในตำแหน่งดัชนีนั้น
- ถ้าค่าดัชนีไม่เป็นจำนวนเต็ม ให้ปัดเศษของค่าดัชนีลงเป็นจำนวนเต็ม และใช้ค่าข้อมูลในตำแหน่งดัชนีที่ปัดเศษแล้ว และตำแหน่งถัดไป คำนวณค่าเฉลี่ยของข้อมูลทั้งสองค่านี้ เพื่อให้ได้ค่าเบอร์เซ็นไทล์ที่  $p$

ค่าเบอร์เซ็นไทล์มีความสำคัญในการวิเคราะห์ข้อมูล ช่วยให้เราเข้าใจการกระจายตัวของข้อมูล วัดความแปรปรวนของข้อมูล และพิจารณาความสัมพันธ์ของข้อมูลในชุดต่าง ๆ นอกจากนี้ค่าเบอร์เซ็นไทล์ยังเป็นเครื่องมือสำคัญในการตัดสินใจเชิงปริมาณอีกด้วย

## 2.2 เทคโนโลยีที่เกี่ยวข้อง

### 2.2.1 ไพทอน (Python)

ไพทอนเป็นภาษาพัฒนาโปรแกรมหนึ่งที่มีการใช้อย่างแพร่หลาย โดยไพทอนถูกปล่อยตัวออกมารุ่นแรก เมื่อปี ค.ศ. 1991 ไพทอนเป็นภาษาการเขียนรหัสโปรแกรมที่ไม่ซับซ้อนและเข้าใจง่าย จึงมักเป็นตัวเลือกแรกๆ ใน การเขียนรหัสโปรแกรมซึ่งถูกนำมาใช้ในงานที่แตกต่างหลากหลายด้วยเช่น การพัฒนาเว็บไซต์, การวิเคราะห์ข้อมูล, การทำงานของเอไอ, การทำงานของเครื่องจักรการเรียนรู้ และการทำงานแบบอัตโนมัติในรูปแบบต่างๆ เป็นต้น ซึ่งในการวิเคราะห์ข้อมูลต่างๆ นั้นไพทอนเป็นภาษาเขียนโค้ดที่ง่ายต่อการวิเคราะห์ข้อมูลต่างๆ ทำให้มีโมดูลช่วยมากมายสำหรับการวิเคราะห์ข้อมูลตัวอย่างเช่น โมดูลสำหรับการจัดการข้อมูล, โมดูลสำหรับการแสดงผลของข้อมูลออกมาเป็นรูปภาพ และโมดูลสำหรับการแสดงผลทางสถิติต่างๆ เป็นต้น ซึ่งโมดูลเหล่านี้สามารถใช้ได้ผ่านทาง library ที่มีคนสร้างขึ้นไว้ library ที่สำคัญต่อการวิเคราะห์ข้อมูลมีดังนี้

1) Numpy: เป็น library สำหรับการคำนวณเชิงตัวเลขโดยสนับสนุนการคำนวณด้านเมทริกซ์, การคำนวณแต่ละแถว และการคำนวณขั้นสูงแบบต่างๆ ของคณิตศาสตร์ เป็นที่แพร่หลายในการใช้งานด้าน linear algebra, statistical analysis และ numerical optimization

2) Pandas: เป็นหนึ่งใน library ที่สำคัญอย่างมากสำหรับจัดการข้อมูลและวิเคราะห์ข้อมูล โดยจะมีฟังก์ชันต่างๆ มากมายในการจัดการข้อมูลตัวอย่างเช่น การจัดการกับข้อมูลที่สูญหาย, การกรองข้อมูล, การเรียงข้อมูล และการรวมข้อมูลเข้าด้วยกันเป็นต้น

3) Matplotlib: เป็น library ในการแสดงข้อมูลผ่านทางภาพ สามารถที่จะพล็อตกราฟรูปแบบต่างๆ ออกมามากมายไม่ว่าจะเป็น กราฟแท่ง, กราฟทรงกลม หรือการพล็อตจุดของข้อมูล

4) Seaborn: เป็น library ที่มีพื้นฐานมาจาก Matplotlib จึงสามารถแสดงภาพข้อมูลออกมาได้อย่างหลากหลายโดยสามารถที่จะใส่ข้อมูลเชิงลึกเข้าไปในรูปภาพต่างๆ ของเราราได้ สามารถปรับแต่งกราฟให้มีสีที่หลากหลายหรือพื้นหลังที่ดูง่ายได้

5) Scikit-learn: เป็น library สำหรับการเรียนรู้ด้วยเครื่องจักร ให้มีความง่ายมากขึ้นและมีโมดูลที่มีประสิทธิภาพต่างๆ สำหรับการวิเคราะห์ข้อมูล โดยตัว library นี้จะมีขั้นตอนวิธีที่หลากหลายสำหรับการเรียนรู้ด้วยเครื่องจักร ในรูปแบบต่างๆ ตัวอย่างเช่น classification, regression, clustering และ dimensionality reduction เป็นต้น

ซึ่งในโครงการนี้ก็จะใช้งาน library เหล่านี้เพื่อความสะดวกในการพัฒนาโปรแกรม

### 2.2.2 Google Colaboratory

Google Colaboratory หรือเรียกว่า Google Colab นั้นเป็นพื้นที่ใช้งานพรีสำหรับการเขียนโค้ดไปท่อน โดยที่สามารถเขียนโค้ด รันโค้ด และดูผลลัพธ์ได้ผ่านเบราว์เซอร์ได้โดยตรง ทำให้เป็นประโยชน์อย่างมากต่อการวิเคราะห์ข้อมูล, machine learning, deep learning, และงานวิจัยอื่นๆ อีกมากมายโดยข้อดีของ Google Colaboratory มีดังต่อไปนี้

- 1) ผู้ใช้งานสามารถเข้าใช้งาน Google Colaboratory ได้โดยไม่ต้องติดตั้งอะไรเพิ่มเติม
- 2) สามารถประมวลผลข้อมูลผ่านทาง Google Colaboratory หมายความว่า Google Colaboratory จัดการให้เข้าถึง CPUs, GPUs (Graphics Processing Units) และ TPUs (Tensor Processing Units) เพื่อใช้งานในการประมวลผลโปรแกรมต่างๆ
- 3) เกิดการทำงานร่วมกันผ่าน Google Colaboratory ให้ผู้ใช้งานสามารถทำงานแบบบีจูบันโดยไม่มีความล่าช้าหรือต้องรอ สามารถนำไปทำงานร่วมกับผู้อื่นได้
- 4) การใช้งานร่วมกับ Google Drive ซึ่งเป็นแหล่งเก็บข้อมูลหรือไฟล์งานรูปแบบต่างๆ จึงสะดวกที่จะเรียกข้อมูลหรือไฟล์งานได้โดยตรง
- 5) มีการติดตั้ง library ที่เป็นที่นิยมให้ก่อนแล้วไม่ว่าจะเป็น library numpy, pandas หรือ matplotlib ก็มีการติดตั้งให้ก่อนแล้ว

## 2.3 งานวิจัยที่เกี่ยวข้อง

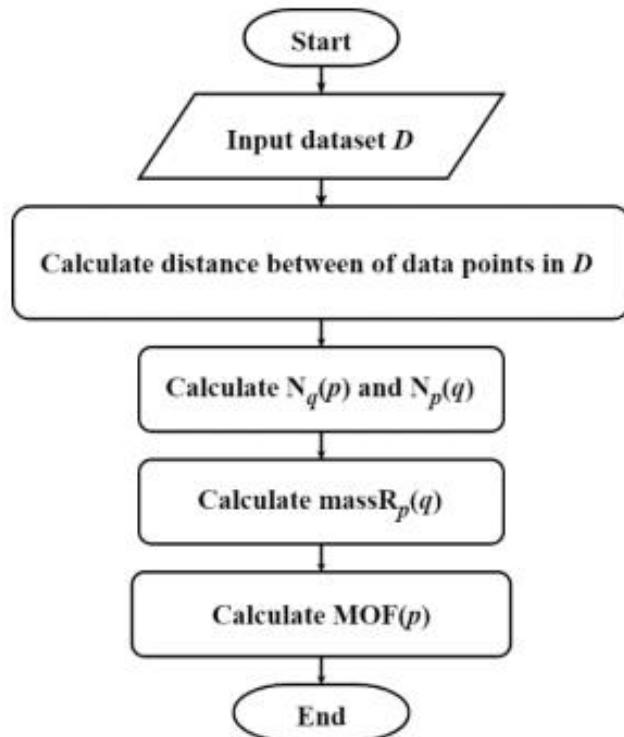
ข้อมูลที่ผิดปกติ (outlier) นั้นเป็นเรื่องที่พบเห็นได้บ่อยในการทำงานด้านสถิติ โดยข้อมูลที่ผิดปกตินั้นสามารถนิยามออกมายield คือเป็นจุดข้อมูลที่แตกต่างจากจุดข้อมูลตัวอื่น ซึ่งข้อมูลที่ปกติส่วนใหญ่แล้วจะถูกห้อมล้อมด้วยจุดข้อมูลมาก many ในทางตรงกันข้ามข้อมูลที่ผิดปกติมักจะอยู่ห่าง หรือไม่ถูกห้อมล้อมด้วยข้อมูลตัวอื่น จากที่กล่าวมานั้นจึงเป็นผลให้มีการศึกษาการตรวจจับข้อมูลผิดปกติเหล่านี้ ซึ่งวิธีการ Mass-ratio-variance Outlier Factor (MOF) ก็เป็นอีกหนึ่งวิธีที่ตรวจจับข้อมูลที่ผิดปกติที่ถูกเสนอขึ้นมา โดยหลักการของ MOF นั้นจะทำการให้คะแนนจุดข้อมูลแต่ละตัวจากการคำนวณความแปรปรวนของ mass-ratio จากจุดข้อมูลกับจุดข้อมูลตัวอื่นๆ โดยจากการการรัศมีวงกลมเพื่อหาค่า MOF นั้นข้อมูลที่ผิดปกตินั้นจะมีจำนวนข้อมูลในรัศมีวงกลมที่ทางออกเพียงเล็กน้อยเมื่อเทียบกับข้อมูลที่ปกติ ดังนั้นค่า mass-ratio ของข้อมูลที่ผิดปกตินั้นจะมีความแตกต่างจากข้อมูลปกติอย่างเห็นได้ชัด การทำงานของขั้นตอนวิธีนี้ทำการคำนวณค่า MOF ออกมาโดยที่ไม่ต้องกำหนดค่าพารามิเตอร์ใดๆ ซึ่งจากการทดสอบบนนั้น MOF algorithm ได้แสดงถึงประสิทธิภาพในการตรวจจับข้อมูลผิดปกติที่สูงเมื่อเทียบกับวิธีการอื่นๆ โดยขั้นตอนวิธีนี้ได้ทดสอบกับข้อมูลสังเคราะห์ซึ่งถูกทดสอบกับขั้นตอนวิธีการตรวจจับข้อมูลแบบอื่นคือ LOF algorithm และ FastABOD algorithm นอกจากนี้ยังทดสอบกับข้อมูลโลจิริ่งมาแล้ว[10]

โดยขั้นตอนวิธีการทำงานของ MOF มีดังนี้

กำหนดให้  $p$  และ  $q$  เป็นจุดข้อมูลที่เราสนใจ

- 1) การคำนวณระยะทางโดยวิธีการยุคลิด เป็นการหาระยะทางระหว่างจุดข้อมูลนำมาใช้เป็นรัศมีในการวงกลมเพื่อนับจุดภายในวงกลมที่ทางออกไปนั้น
- 2) หาดวงกลมที่มีรัศมีคือระยะทางระหว่างจุดจากจุดข้อมูลที่ทำการคำนวณค่าระยะทางก่อนหน้า แล้วนับจำนวนข้อมูลภายในรัศมีวงกลมที่ทางออก
- 3) หาค่า mass-ratio ( $\text{massR}_p(q)$ ) วิธีการคือการนำจำนวนจุดข้อมูลที่เราสนใจ ( $N_p(q)$ ) จากขั้นตอนที่ 2 หารด้วยจำนวนจุดที่ทางรัศมีออกของข้อมูลอีกจุดนึง ( $N_q(p)$ ) ที่ทำการคำนวณค่าระยะทางแบบยุคลิด
- 4) คำนวณขั้นตอนที่ 1 ถึง 3 เพื่อหาค่า mass-ratio ทั้งหมดของจุดที่เราสนใจกับทุกๆ จุดข้อมูล
- 5) คำนวณ MOF โดยวิธีการคือหาค่าความแปรปรวนของ mass-ratio ของจุดที่เราสนใจกับทุกจุดข้อมูลอื่นๆ

จากขั้นตอนทั้งหมดเราจะสามารถหาค่า MOF ออกมาได้โดยถ้าค่า MOF ที่จุดใดสูงก็มีแนวโน้มที่จะเป็นจุดข้อมูลผิดปกตินั่นเอง



ภาพที่ 2.1 แผนภาพการทำงานของ MOF algorithm[10]

## บทที่ 3

### วิธีการดำเนินงาน

จากความต้องการวิเคราะห์การเกาะกลุ่มข้อมูลโดยใช้ MOF algorithm ช่วยในการกำจัดข้อมูลผิดปกติ แล้วดูประสิทธิภาพการจัดกลุ่มข้อมูลแบบไม่กำจัดข้อมูลผิดปกติกับแบบกำจัดข้อมูลผิดปกติ โดยอย่างที่จะทดสอบ กับข้อมูลสังเคราะห์รูปประจำต่างๆ และข้อมูลโลกความเป็นจริงก็คือ ชุดข้อมูลจาก spotify ผู้จัดทำได้ดำเนินการ ดังต่อไปนี้

- ขั้นตอนการดำเนินงาน
- ข้อมูลที่ใช้ในโครงการ
- เครื่องมือที่ใช้ในการวิเคราะห์การเกาะกลุ่ม
- การวิเคราะห์ข้อมูลและการนำเสนอ
- เทคนิคที่นำมาใช้ในการจัดกลุ่มและวิเคราะห์ข้อมูล

#### 3.1 ประชากรและกลุ่มตัวอย่าง

ประชากร: ข้อมูลของโครงการที่จะนำมาใช้ในโครงการนี้ก็คือ ข้อมูล Spotify และข้อมูลสังเคราะห์รูปประจำต่างๆ จาก Scikit-learn

กลุ่มตัวอย่าง: ใช้เทคนิค systematic sampling เพื่อทำการสุ่มตัวอย่างข้อมูลจาก Spotify เพื่อนำมาใช้ เป็นกลุ่มตัวอย่าง

#### 3.2 เครื่องมือที่ใช้ในการวิเคราะห์การเกาะกลุ่มข้อมูล

ใช้งาน Google colab เพื่อเขียนโค๊ดและใช้ฟังก์ชันต่างๆ จาก Google colab เพื่อแสดงผลลัพธ์จากการ ตรวจสอบข้อมูลผิดปกติและการจัดกลุ่มข้อมูลในรูปแบบต่างๆ เช่น กราฟ ข้อมูลเชิงสถิติ การประเมินการจัดกลุ่ม เป็นต้น

#### 3.3 การเก็บรวบรวมข้อมูล

ผู้จัดทำโครงการสังเคราะห์ข้อมูลจากฟังก์ชันการทำงานของ scikit-learn และนำข้อมูล Spotify จาก เว็บไซต์ Kaggle ซึ่งเป็นเว็บไซต์สำหรับเก็บข้อมูลต่างๆ

### 3.4 การวิเคราะห์ข้อมูลและการนำเสนอ

1. การวิเคราะห์ข้อมูล เมื่อจัดหาข้อมูลสังเคราะห์และนำข้อมูล Spotify จาก Kaggle แล้วก็ทำการตรวจจับข้อมูลผิดปกติจากอัลกอริทึม MOF และทำการจัดกลุ่มข้อมูลที่กำจัดข้อมูลผิดปกติแล้วด้วยอัลกอริทึม k-mean และ agglomerative single linkage นำผลลัพธ์การจัดกลุ่มมาวิเคราะห์ข้อมูลเบรียบโดยวิเคราะห์จากค่า silhouette การแสดงผลทางกราฟ และข้อมูลจากการจัดกลุ่ม

#### 2. การนำเสนอข้อมูล มีดังนี้

2.1 นำเสนอในรูปของตาราง โดยกรอกข้อมูลผลลัพธ์จากการจัดกลุ่มข้อมูลออกมาโดยแบ่งเป็นแนวตั้ง (Column) และแนวนอน (Row) เพื่อจัดข้อมูลให้เป็นระเบียบ

2.2 นำเสนอในรูปกราฟ เพื่อแสดงข้อมูลที่ทำให้เกิดความน่าสนใจ ทำให้อ่านและตีความข้อมูลได้ง่ายและรวดเร็วยิ่งขึ้น

2.3 นำเสนอในรูปของบทความ ลักษณะการเสนอเป็นบทความสั้นๆเพื่ออธิบายตัวข้อมูล และนำเสนอคู่กับข้อมูลตัวเลข

### 3.5 เทคนิคที่นำมาใช้ในการจัดกลุ่มและวิเคราะห์ข้อมูล

ผู้จัดทำโครงงานมีเทคนิคที่นำมาใช้ต่างๆดังต่อไปนี้

1) ทำการสังเคราะห์ข้อมูลรูป่างต่างๆจาก scikit-learn และแสดงผลตัวอย่างข้อมูล Spotify เพื่อดูลักษณะของข้อมูลเพื่อที่จะเลือกลักษณะของข้อมูลที่จะนำมาจัดกลุ่ม

2) ทำการสุ่มตัวอย่างข้อมูลด้วยวิธีการ systematic sampling เนื่องจากข้อมูลที่มีจำนวนมากเกินไปไม่สามารถจะให้คอมพิวเตอร์คำนวณได้หมด

3) ทำการเรียงข้อมูลใหม่แล้วทำการกำจัดข้อมูลผิดปกติด้วยอัลกอริทึม MOF ที่ตั้งเกณฑ์การตรวจจับด้วยค่าเปอร์เซ็นต์

4) นำข้อมูลที่กำจัดข้อมูลผิดปกติแล้วทำการสเกลข้อมูลด้วยวิธีการ min-max scaling เพื่อให้ข้อมูลมีสเกลที่ใกล้เคียงกันไม่ให้เกิดการอคติต่อการจัดกลุ่มข้อมูล

5) หลังจากกำจัดข้อมูลเสรีจแล้วก็ทำการทดสอบกับข้อมูลสังเคราะห์และข้อมูล Spotify เพื่อดูประสิทธิ์ของการทำงานร่วมกันระหว่างอัลกอริทึม MOF และอัลกอริทึมการจัดกลุ่มข้อมูลได้แก่ k-mean และ agglomerative single linkage

6) แสดงผลลัพธ์ออกมาทั้งในรูปของกราฟที่แสดงข้อมูลที่ถูกจัดกลุ่มออกมา วัดกราฟแสดงข้อมูลการจัดกลุ่มของข้อมูลต่างๆตามลักษณะที่เลือก และหาค่า Silhouette เพื่อใช้ในการประเมินประสิทธิภาพของการจัดกลุ่ม

7) ประเมินการจัดกลุ่มแต่ละประเภททั้งการจัดกลุ่มแบบไม่กำจัดข้อมูลผิดปกติและการจัดกลุ่มข้อมูลแบบกำจัดข้อมูลผิดปกติกับข้อมูลสังเคราะห์และข้อมูล Spotify โดยประเมินจากลักษณะการจัดกลุ่มจากการดูค่า Silhouette ประกอบ

8) ตีความและสร้างข้อสรุป

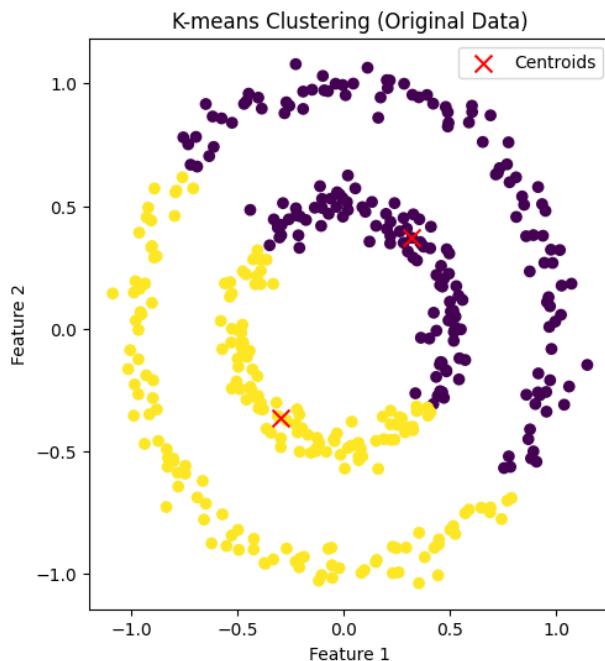
## บทที่ 4

### ผลการวิเคราะห์

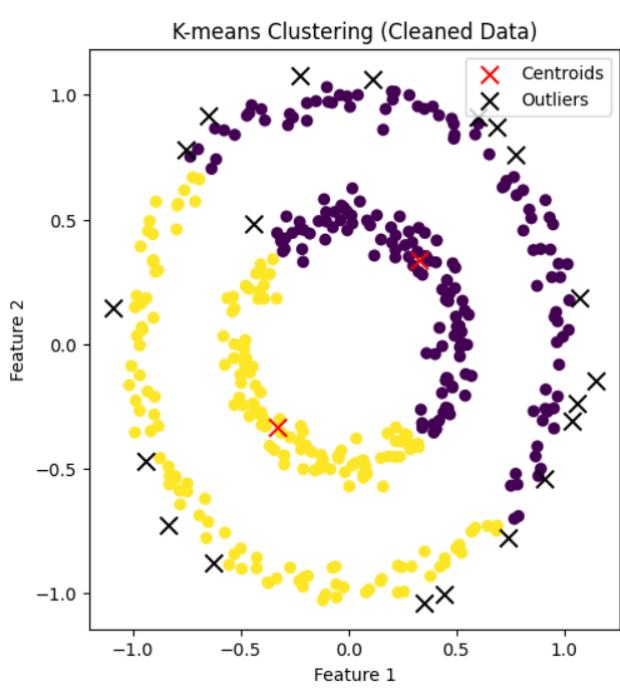
ผู้จัดทำโครงการได้ดำเนินการวิเคราะห์การจัดกลุ่มข้อมูล โดยใช้การตรวจจับข้อมูลผิดปกติผ่านทางอัลกอริทึม MOF ตั้งเกณฑ์การตรวจจับข้อมูลผิดปกติไว้ที่ 95 เปอร์เซ็นไทล์ และจัดกลุ่มข้อมูลโดยวิธีการ K-mean และวิธีการ Agglomerative single linkage ทดสอบกับข้อมูลสังเคราะห์รูปร่าง 5 แบบ ได้แก่ (1) วงกลมช้อนสองวง (2) รูปพระจันทร์ช้อนสองวง (3) กลุ่มข้อมูลกระจายสามกลุ่ม (4) กลุ่มข้อมูลที่เรียงเป็นเส้นสามเส้น (5) กลุ่มข้อมูลที่กระจายเป็นกลุ่มใหญ่ แต่ละรูปร่างมีขนาดข้อมูล 400 ตัว และทดสอบกับชุดข้อมูล Spotify ขนาดข้อมูล 8,000 ตัว ซึ่งมาจากการสุ่มแบบ systematic sampling จากข้อมูลทั้งหมด 42,306 ตัว ผลการวิเคราะห์ทั้งหมดมีดังต่อไปนี้

#### 4.1 ผลการวิเคราะห์กับข้อมูลสังเคราะห์

##### 4.1.1 วิธีการจัดกลุ่ม K-mean



ภาพที่ 4.1.1 การจัดกลุ่มข้อมูลรูปร่างวงกลมช้อนสองวงโดยวิธี k-mean ก่อนการกำจัดข้อมูลผิดปกติ

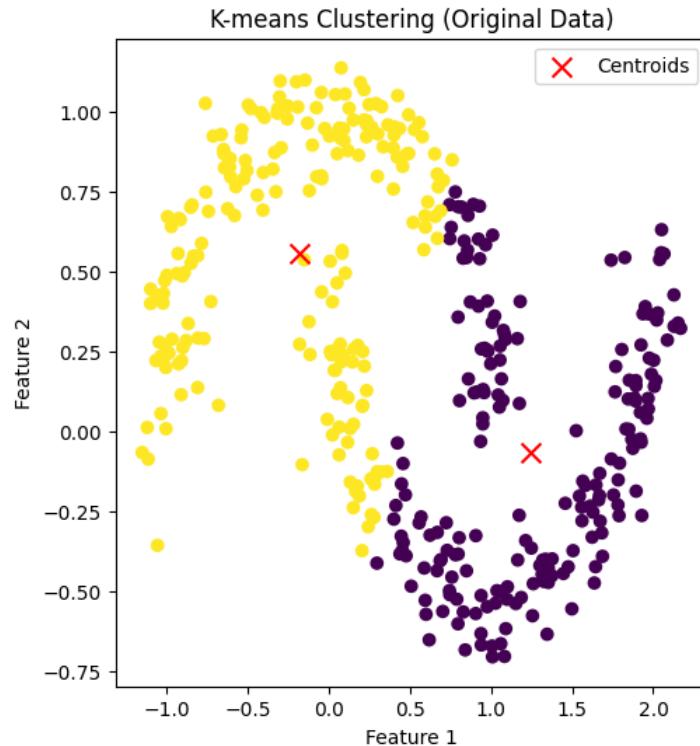


ภาพที่ 4.1.2 การจัดกลุ่มข้อมูลรูปร่างวงกลมช่อนสองวงโดยวิธี k-mean หลังการกำจัดข้อมูลผิดปกติ

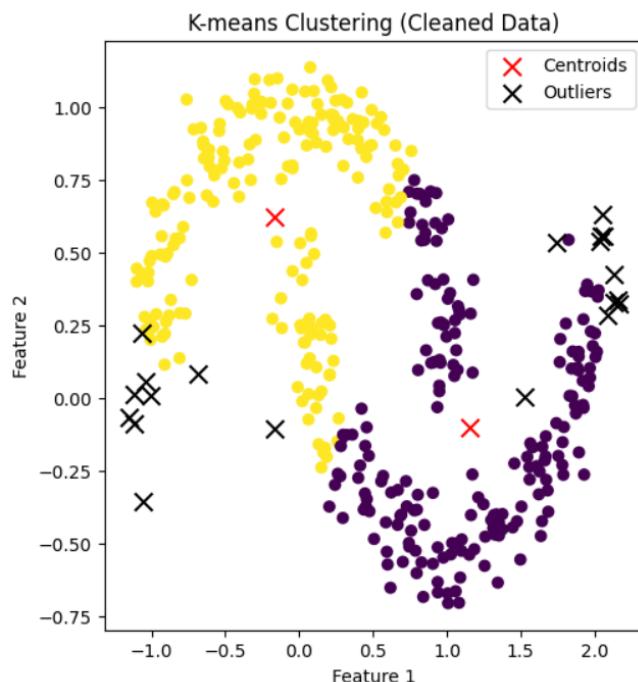
ตารางที่ 4.1.1 แสดงคะแนนซิลูเอต (Silhouette) ก่อนกำจัดข้อมูลผิดปกติและหลังกำจัดข้อมูลผิดปกติของข้อมูลรูปร่างวงกลมช่อนสองวง

สถานะข้อมูล	คะแนนซิลูเอต (Silhouette)
ก่อนกำจัดข้อมูลผิดปกติ	0.3545
หลังกำจัดข้อมูลผิดปกติ	0.3579

จากตารางที่ 4.1.1 จะเห็นว่าคะแนนซิลูเอตหลังกำจัดข้อมูลผิดปกติมีค่ามากกว่าก่อนกำจัดข้อมูลผิดปกติอยู่ 0.0034



ภาพที่ 4.1.3 การจัดกลุ่มข้อมูลรูปร่างรูปพระจันทร์ช้อนสองวงโดยวิธี k-mean ก่อนกำจัดข้อมูลผิดปกติ

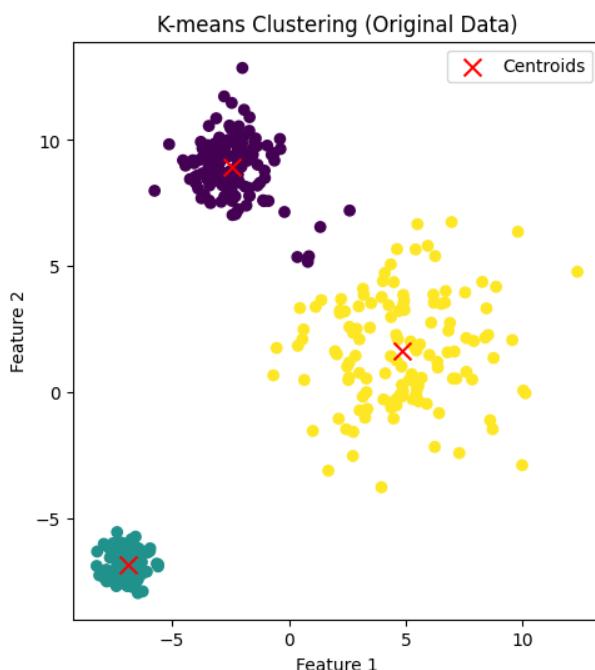


ภาพที่ 4.1.4 การจัดกลุ่มข้อมูลรูปร่างรูปพระจันทร์ช้อนสองวงโดยวิธี k-mean หลังกำจัดข้อมูลผิดปกติ

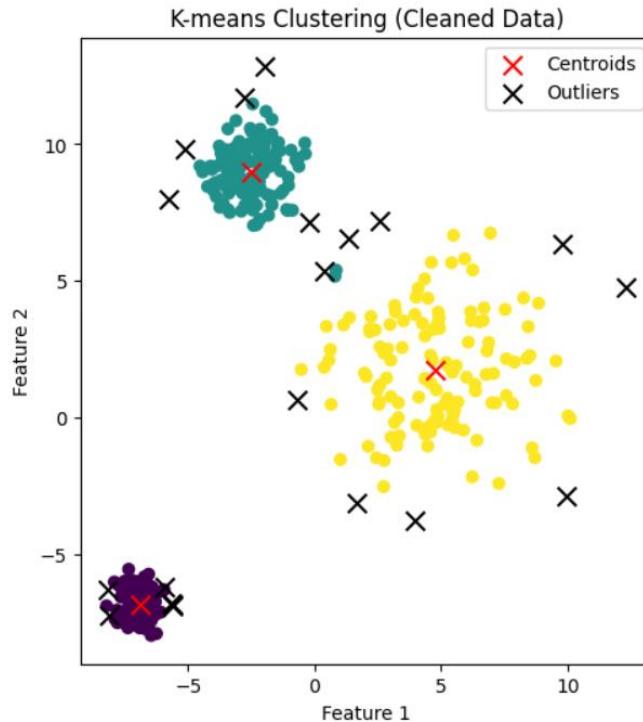
ตารางที่ 4.1.2 แสดงคะแนนชิลูเอต (Silhouette) ก่อนกำจัดข้อมูลผิดปกติและหลังกำจัดข้อมูลผิดปกติของข้อมูลรูปร่างพระจันทร์ซ้อนสองวง

สถานะข้อมูล	คะแนนชิลูเอต (Silhouette)
ก่อนกำจัดข้อมูลผิดปกติ	0.4831
หลังกำจัดข้อมูลผิดปกติ	0.4854

จากตารางที่ 4.1.2 จะเห็นว่าคะแนนชิลูเอตหลังกำจัดข้อมูลผิดปกติมีค่ามากกว่าก่อนกำจัดข้อมูลผิดปกติอยู่ 0.0023



ภาพที่ 4.1.5 การจัดกลุ่มข้อมูลรูปร่างก้นกลุ่มข้อมูลกระจายสามกลุ่มโดยวิธี k-mean ก่อนกำจัดข้อมูลผิดปกติ

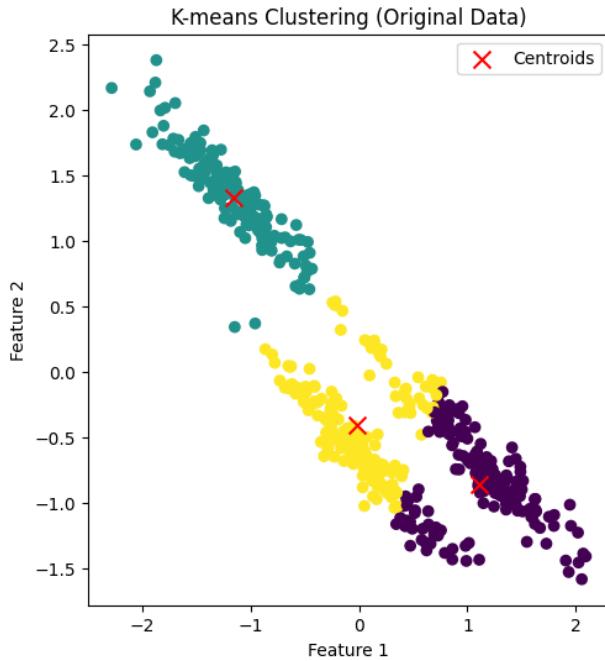


ภาพที่ 4.1.6 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลกระจายสามกลุ่มโดยวิธี k-mean หลังกำจัดข้อมูลผิดปกติ

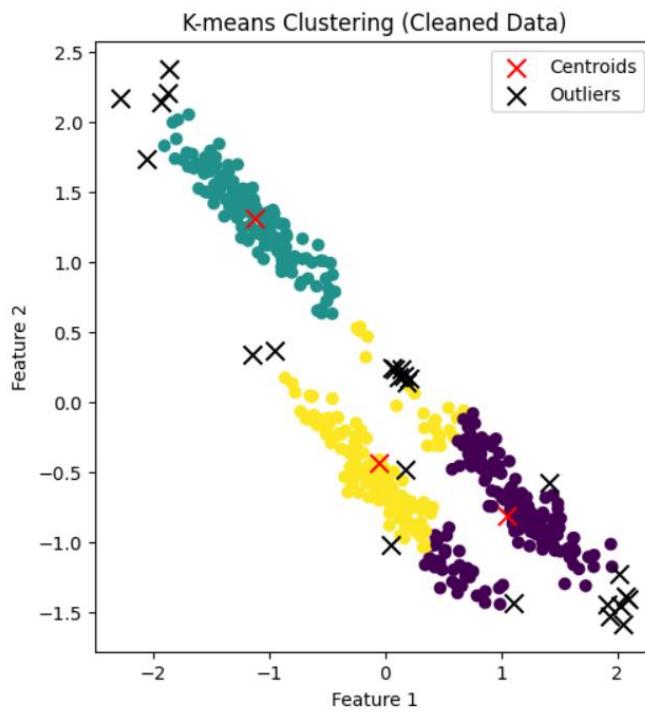
ตารางที่ 4.1.3 แสดงคะแนนชิลูเอตก่อนกำจัดข้อมูลผิดปกติและหลังกำจัดข้อมูลผิดปกติของข้อมูลรูปร่างกลุ่มข้อมูลกระจายสามกลุ่ม

สถานะข้อมูล	คะแนนชิลูเอต (Silhouette)
ก่อนกำจัดข้อมูลผิดปกติ	0.7824
หลังกำจัดข้อมูลผิดปกติ	0.8006

จากตารางที่ 4.1.3 จะเห็นว่าคะแนนชิลูเอตหลังกำจัดข้อมูลผิดปกติมีค่ามากกว่าก่อนกำจัดข้อมูลผิดปกติอยู่ 0.0182



ภาพที่ 4.1.7 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลที่เรียกเป็นเส้นสามเหลี่ยมโดยวิธี k-mean ก่อนกำจัดข้อมูลผิดปกติ

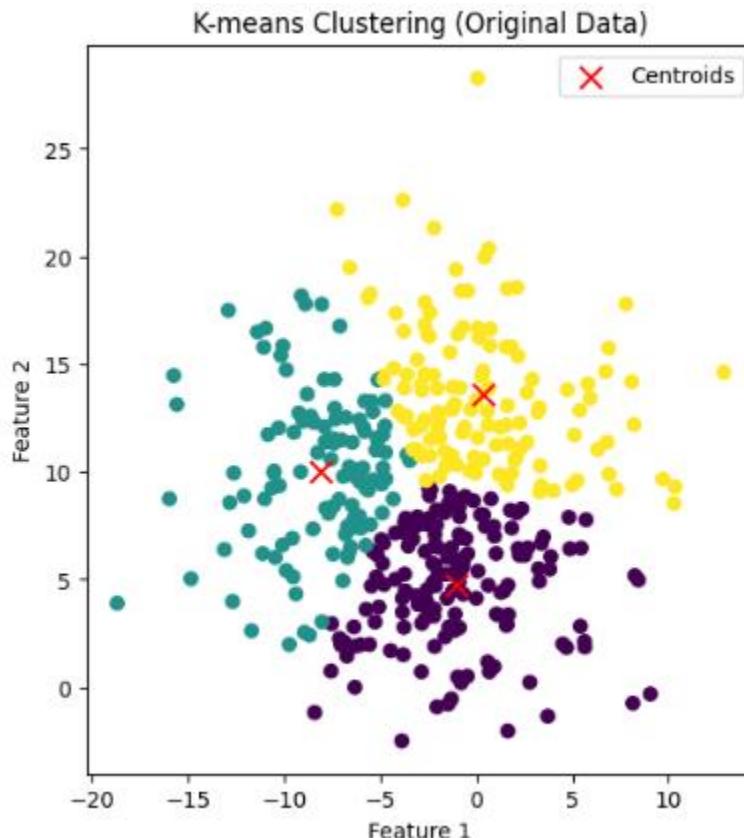


ภาพที่ 4.1.8 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลที่เรียกเป็นเส้นสามเหลี่ยมโดยวิธี k-mean หลังกำจัดข้อมูลผิดปกติ

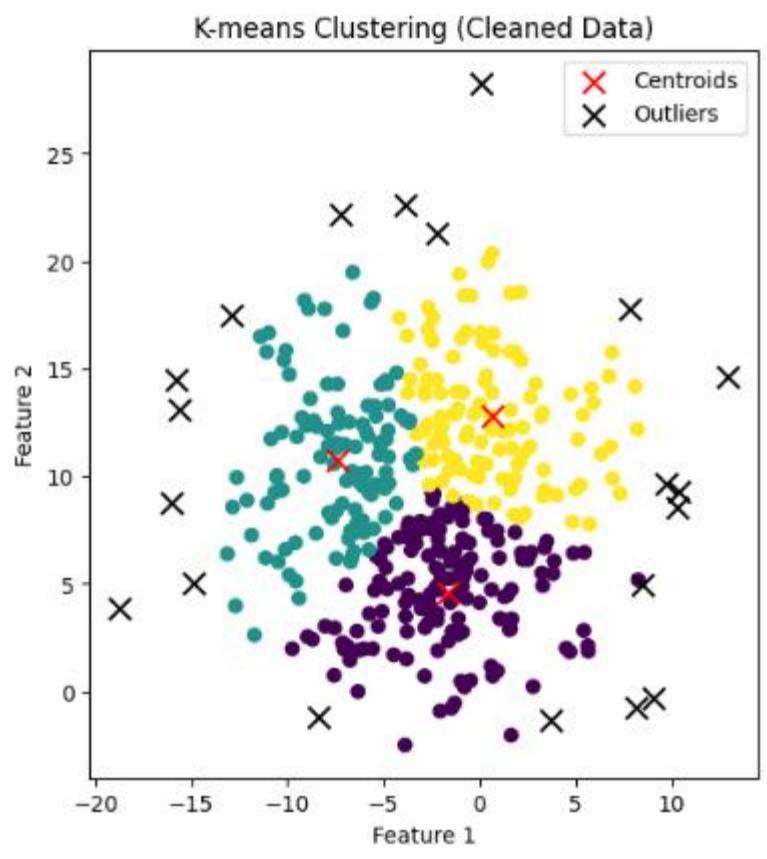
ตารางที่ 4.1.4 แสดงคะแนนชิลูเอตก่อนกำจัดข้อมูลผิดปกติและหลังกำจัดข้อมูลผิดปกติของข้อมูลรุ่ปร่างกลุ่มข้อมูลที่เรียงเป็นเส้นสามเส้น

สถานะข้อมูล	คะแนนชิลูเอต (Silhouette)
ก่อนกำจัดข้อมูลผิดปกติ	0.5428
หลังกำจัดข้อมูลผิดปกติ	0.5566

จากตารางที่ 4.1.4 จะเห็นว่าคะแนนชิลูเอตหลังกำจัดข้อมูลผิดปกติมีค่ามากกว่าก่อนกำจัดข้อมูลผิดปกติอยู่ 0.0138



ภาพที่ 4.1.9 การจัดกลุ่มข้อมูลรุ่ปร่างกลุ่มข้อมูลที่กระจายเป็นกลุ่มใหญ่โดยวิธี k-mean ก่อนกำจัดข้อมูลผิดปกติ



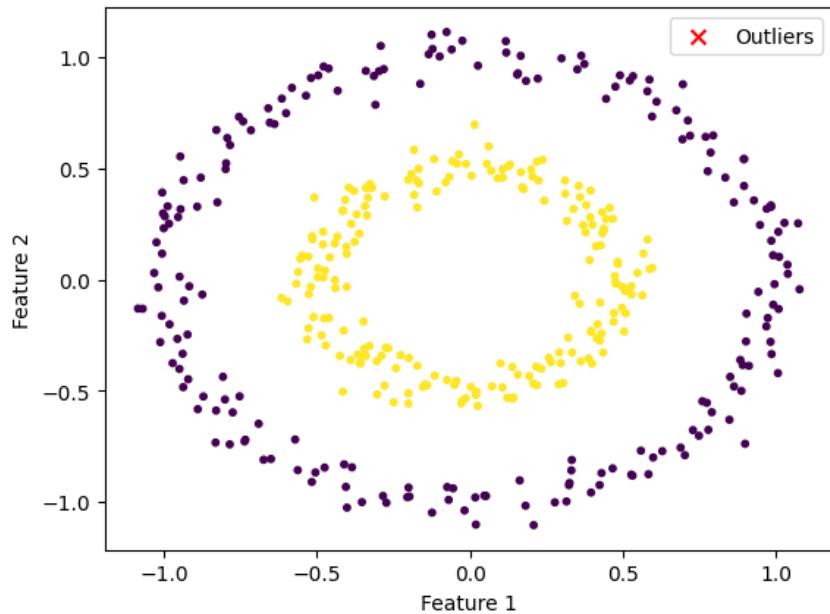
ภาพที่ 4.1.10 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลที่กระจายเป็นกลุ่มใหญ่โดยวิธี k-mean หลังกำจัดข้อมูลผิดปกติ

ตารางที่ 4.1.5 แสดงคะแนนชิลูเอตก่อนกำจัดข้อมูลผิดปกติและหลังกำจัดข้อมูลผิดปกติของข้อมูลรูปร่างกลุ่มข้อมูลที่กระจายเป็นกลุ่มใหญ่

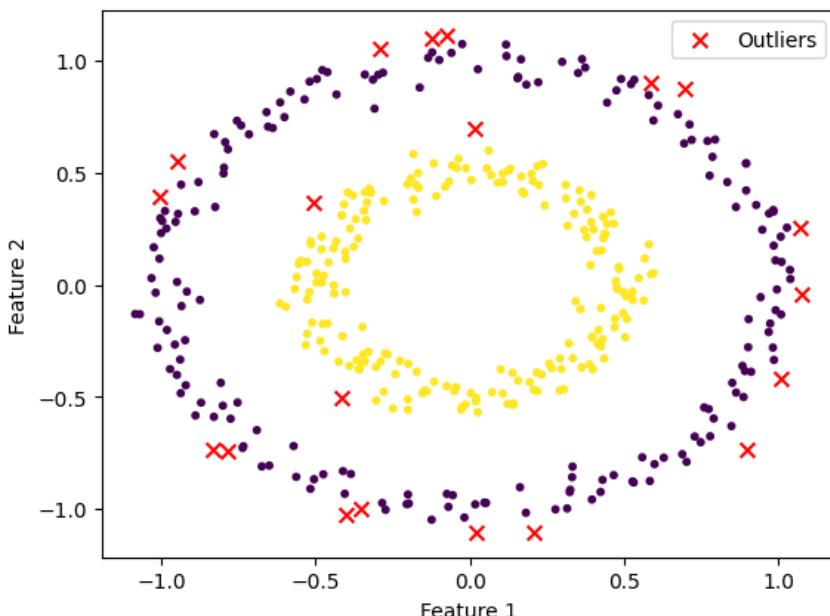
สถานะข้อมูล	คะแนนชิลูเอต (Silhouette)
ก่อนกำจัดข้อมูลผิดปกติ	0.3480
หลังกำจัดข้อมูลผิดปกติ	0.3570

จากตารางที่ 4.1.5 จะเห็นว่าคะแนนชิลูเอตหลังกำจัดข้อมูลผิดปกติมีค่ามากกว่าก่อนกำจัดข้อมูลผิดปกติอยู่ 0.009

#### 4.1.1 วิธีการจัดกลุ่มแบบ Agglomerative single linkage



ภาพที่ 4.1.11 การจัดกลุ่มข้อมูลรูปร่างวงกลมซ้อนสองวงโดยวิธี Agglomerative ก่อนการกำจัดข้อมูลผิดปกติ

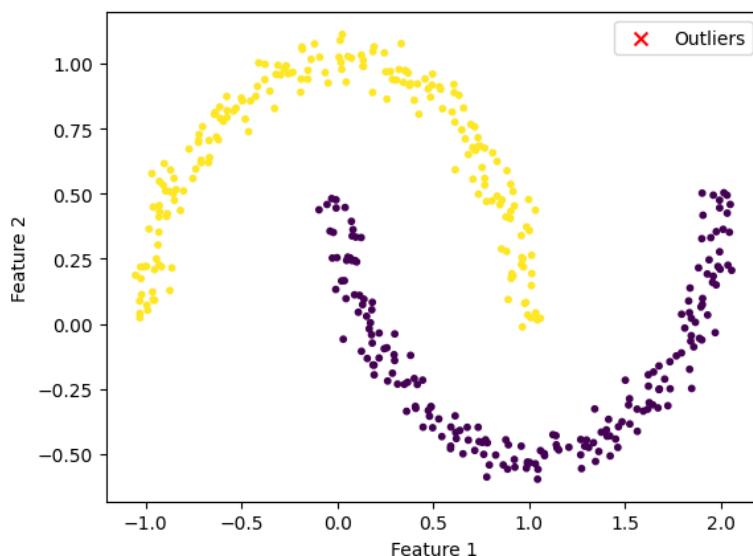


ภาพที่ 4.1.12 การจัดกลุ่มข้อมูลรูปร่างวงกลมซ้อนสองวงโดยวิธี Agglomerative หลังการกำจัดข้อมูลผิดปกติ

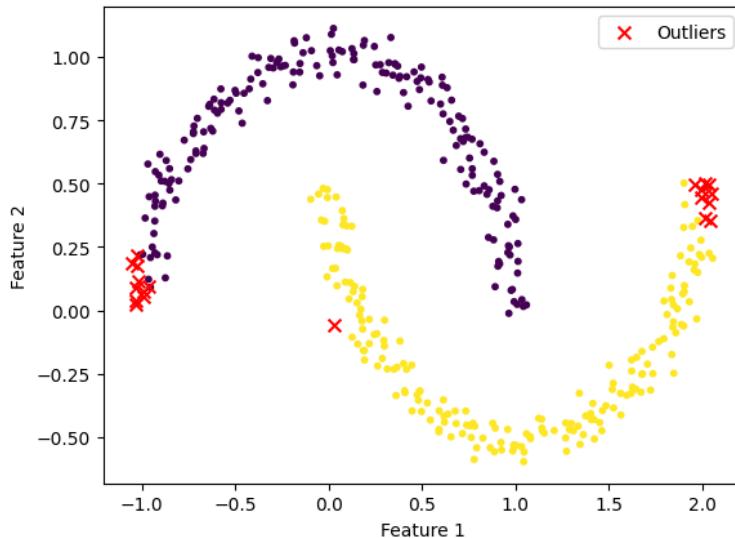
ตารางที่ 4.1.6 แสดงคะแนนชิลูเอตtruปร่างวงกลมซ้อนสองวงของวิธี Agglomerative ก่อนกำจัดข้อมูลผิดปกติและหลังกำจัดข้อมูลผิดปกติ

สถานะข้อมูล	คะแนนชิลูเอต (Silhouette)
ก่อนกำจัดข้อมูลผิดปกติ	0.1122
หลังกำจัดข้อมูลผิดปกติ	0.1220

จากตารางที่ 4.1.6 จะเห็นว่าคะแนนชิลูเอตหลังกำจัดข้อมูลผิดปกติมีค่ามากกว่าก่อนกำจัดข้อมูลผิดปกติอยู่ 0.2342



ภาพที่ 4.1.13 การจัดกลุ่มข้อมูลรูปร่างพระจันทร์ซ้อนสองวงโดยวิธี Agglomerative ก่อนการกำจัดข้อมูลผิดปกติ

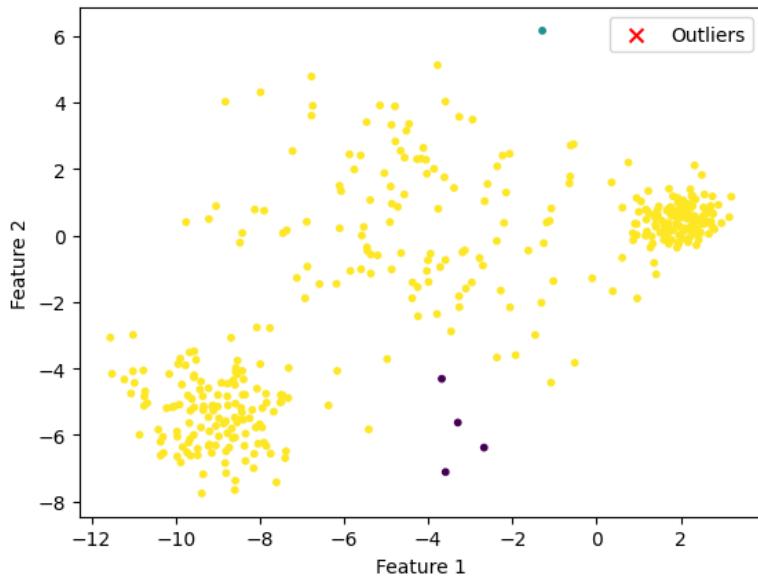


ภาพที่ 4.1.14 การจัดกลุ่มข้อมูลรูปร่างพระจันทร์ซ้อนสองวงโดยวิธี Agglomerative หลังการกำจัดข้อมูลผิดปกติ

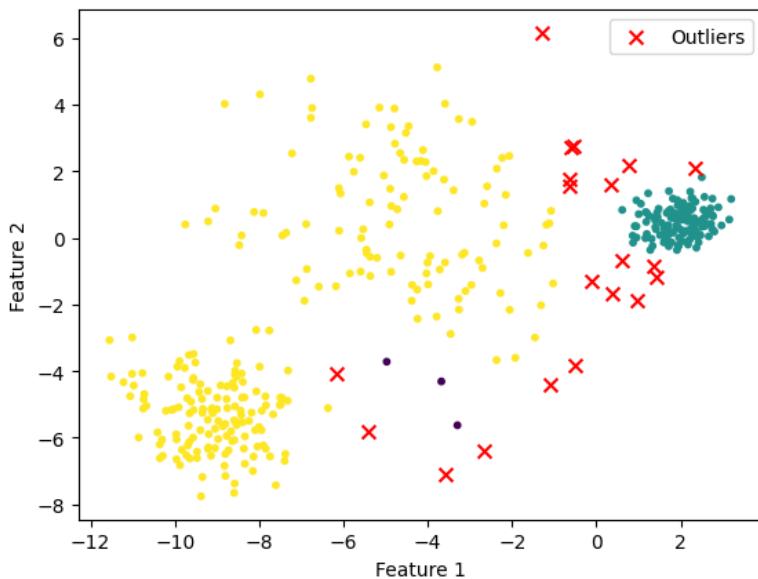
ตารางที่ 4.1.7 แสดงคะแนนชิลูเอตจากการจัดกลุ่มข้อมูลรูปร่างพระจันทร์ซ้อนสองวงโดยวิธี Agglomerative ก่อนกำจัดข้อมูลผิดปกติและหลังกำจัดข้อมูลผิดปกติ

สถานะข้อมูล	คะแนนชิลูเอต (Silhouette)
ก่อนกำจัดข้อมูลผิดปกติ	0.3308
หลังกำจัดข้อมูลผิดปกติ	0.3357

จากตารางที่ 4.1.7 จะเห็นว่าคะแนนชิลูเอตหลังกำจัดข้อมูลผิดปกติมีค่ามากกว่าก่อนกำจัดข้อมูลผิดปกติอยู่ 0.0049



ภาพที่ 4.1.15 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลกระจายสามกลุ่ม ก่อนการกำจัดข้อมูลผิดปกติ

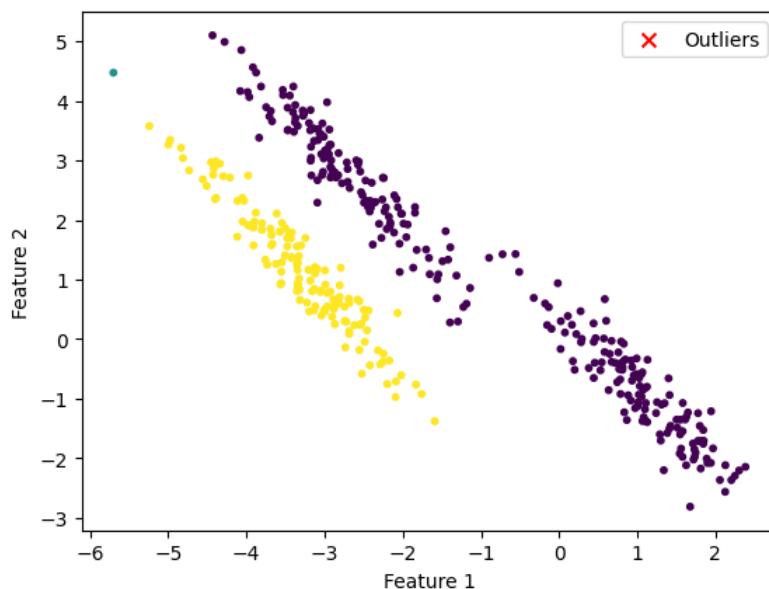


ภาพที่ 4.1.16 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลกระจายสามกลุ่ม หลังการกำจัดข้อมูลผิดปกติ

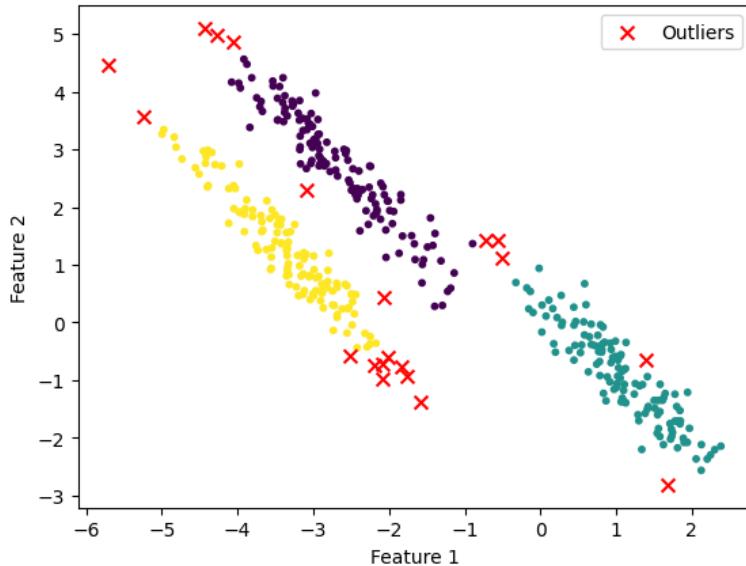
ตารางที่ 4.1.8 แสดงคะแนนชิลูเอตจากการจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลระยะยาวสามกลุ่ม โดยวิธี Agglomerative ก่อนกำจัดข้อมูลผิดปกติและหลังกำจัดข้อมูลผิดปกติ

สถานะข้อมูล	คะแนนชิลูเอต (Silhouette)
ก่อนกำจัดข้อมูลผิดปกติ	-0.1165
หลังกำจัดข้อมูลผิดปกติ	0.3301

จากตารางที่ 4.1.8 จะเห็นว่าคะแนนชิลูเอตหลังกำจัดข้อมูลผิดปกติมีค่ามากกว่าก่อนกำจัดข้อมูลผิดปกติอยู่ 0.4466



ภาพที่ 4.1.17 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลที่เรียงเป็นเส้นสามเส้น ก่อนการกำจัดข้อมูลผิดปกติ

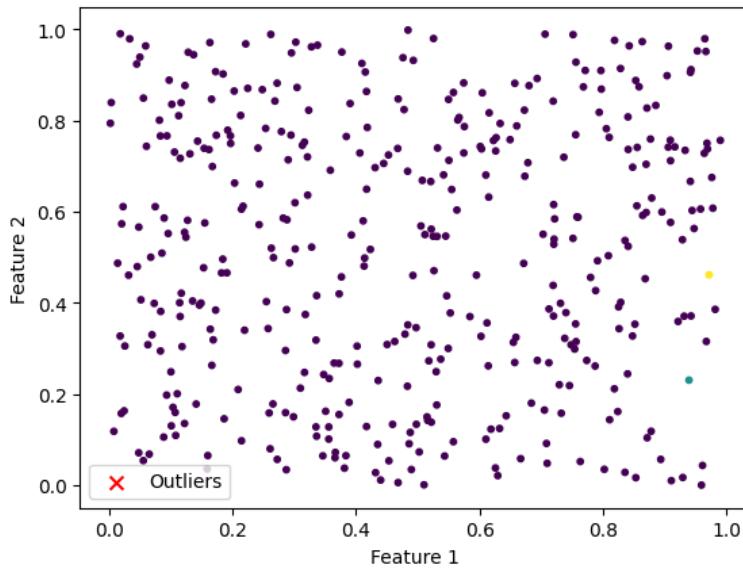


ภาพที่ 4.1.18 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลที่เรียกเป็นเส้นสามเส้น หลังการกำจัดข้อมูลผิดปกติ

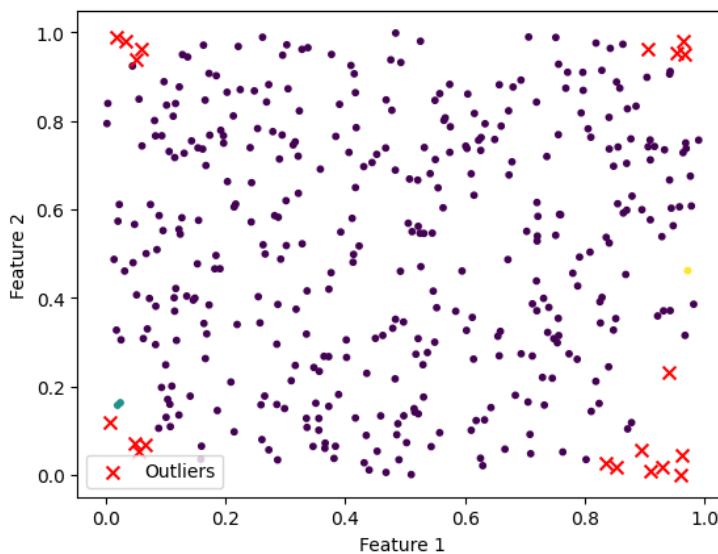
ตารางที่ 4.1.9 แสดงคะแนนชิลูเอต จากการจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลที่เรียกเป็นเส้นสามเส้น โดยวิธี Agglomerative ก่อนกำจัดข้อมูลผิดปกติและหลังกำจัดข้อมูลผิดปกติ

สถานะข้อมูล	คะแนนชิลูเอต (Silhouette)
ก่อนกำจัดข้อมูลผิดปกติ	0.1529
หลังกำจัดข้อมูลผิดปกติ	0.4928

จากตารางที่ 4.1.9 จะเห็นว่าคะแนนชิลูเอตหลังกำจัดข้อมูลผิดปกติมีค่ามากกว่าก่อนกำจัดข้อมูลผิดปกติอยู่ 0.3399



ภาพที่ 4.1.19 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลที่กระจายเป็นกลุ่มใหญ่ก่อนการกำจัดข้อมูลผิดปกติ



ภาพที่ 4.1.20 การจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลที่กระจายเป็นกลุ่มใหญ่หลังการกำจัดข้อมูลผิดปกติ

ตารางที่ 4.1.10 แสดงคะแนนซิลูอ็อต จากการจัดกลุ่มข้อมูลรูปร่างกลุ่มข้อมูลที่กระจายเป็นกลุ่มใหญ่ โดยวิธี Agglomerative ก่อนกำจัดข้อมูลผิดปกติและหลังกำจัดข้อมูลผิดปกติ

สถานะข้อมูล	คะแนนซิลูอ็อต (Silhouette)
ก่อนกำจัดข้อมูลผิดปกติ	-0.0082
หลังกำจัดข้อมูลผิดปกติ	-0.1610

จากตารางที่ 4.1.10 จะเห็นว่าคะแนนชิลล์อตหลังกำจัดข้อมูลพิດปกติมีค่าเฉลี่ยกว่าก่อนกำจัดข้อมูล พิດปกติอยู่ 0.1528

## 4.2 ผลการวิเคราะห์ข้อมูล spotify

### 4.2.1 ผลการตรวจสอบข้อมูล spotify ก่อนและหลังกำจัดข้อมูลพิດปกติ

ตารางที่ 4.2.1 แสดงผลข้อมูลทั้งหมดของ spotify ตามลักษณะที่จะนำมาใช้

	Daceability	Energy	Loudness	Speechiness	Acousticness
Count	42305	42305	42305	42305	42305
Mean	0.6394	0.7625	-6.4654	0.1366	0.0962
Std	0.1566	0.1838	2.9412	0.1262	0.1708

ตารางที่ 4.2.2 แสดงผลข้อมูลทั้งหมดของ spotify ตามลักษณะที่จะนำมาใช้ (ต่อ)

	Instrumentalness	Liveness	Valence	Tempo
Count	42305	42305	42305	42305
Mean	0.2831	0.2141	0.3571	147.4740
Std	0.3708	0.1756	0.2332	23.8446

ตารางที่ 4.2.3 แสดงผลข้อมูลสุ่มตัวอย่างของ spotify ตามลักษณะที่จะนำมาใช้

	Daceability	Energy	Loudness	Speechiness	Acousticness
Count	8000	8000	8000	8000	8000
Mean	0.6474	0.7559	-6.5933	0.1383	0.0976
Std	0.1547	0.1849	2.986	0.1288	0.1698

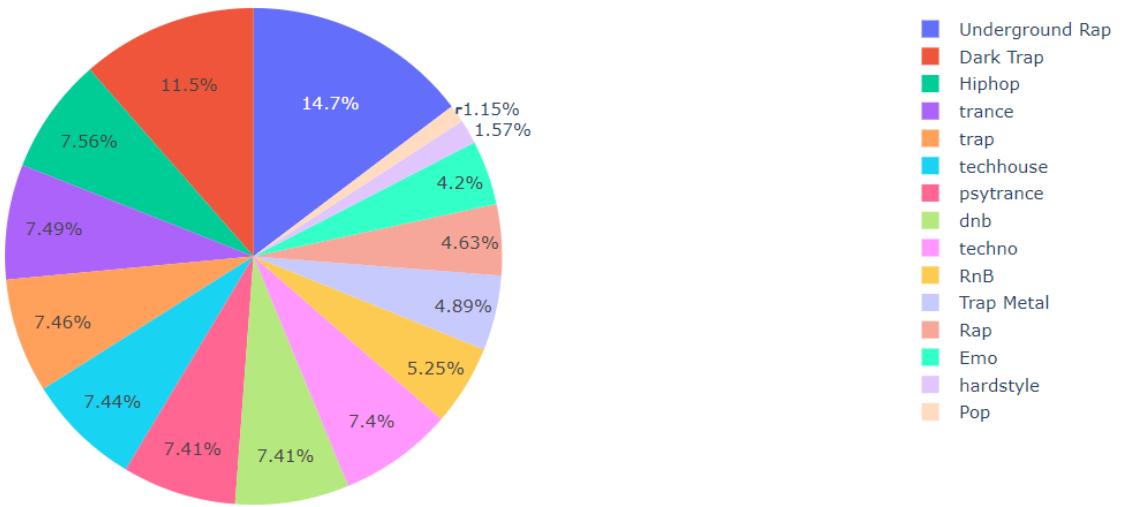
ตารางที่ 4.2.4 แสดงผลข้อมูลสุ่มตัวอย่างของ spotify ตามลักษณะที่จะนำมาใช้ (ต่อ)

	Instrumentalness	Liveness	Valence	Tempo
Count	8000	8000	8000	8000
Mean	0.2938	0.2098	0.3665	147.7066
Std	0.3763	0.1718	0.2364	24.5973

จากตารางที่ 4.2.1 ถึงตารางที่ 4.2.4 ข้อมูลจากการสุ่มตัวอย่างและข้อมูลทั้งหมด ลักษณะ Daceability มีค่าเฉลี่ยต่างกันร้อยละ 1.25 และค่าความแปรปรวนต่างกันร้อยละ 1.21 ลักษณะ Energy มีค่าเฉลี่ยต่างกันร้อยละ 0.86 และค่าความแปรปรวนต่างกันร้อยละ 0.6 ลักษณะ Loudness มีค่าเฉลี่ยต่างกันร้อยละ 1.98 และค่าความแปรปรวนต่างกันร้อยละ 1.52 ลักษณะ Speechiness มีค่าเฉลี่ยต่างกันร้อยละ 1.24 และค่าความแปรปรวนต่างกันร้อยละ 2.06 ลักษณะ Acousticness มีค่าเฉลี่ยต่างกันร้อยละ 1.45 และค่าความแปรปรวนต่างกันร้อยละ 0.58 ลักษณะ Instrumentalness มีค่าเฉลี่ยต่างกันร้อยละ 3.78 และค่าความแปรปรวนต่างกันร้อยละ 1.48 ลักษณะ Liveness มีค่าเฉลี่ยต่างกันร้อยละ 2.01 และค่าความแปรปรวนต่างกันร้อยละ 2.16 ลักษณะ Valence มีค่าเฉลี่ยต่างกันร้อยละ 2.63 และค่าความแปรปรวนต่างกันร้อยละ 1.37 และลักษณะ Tempo มีค่าเฉลี่ยต่างกันร้อยละ 0.16 และค่าความแปรปรวนต่างกันร้อยละ 3.16

#### ตารางที่ 4.2.5 แสดงจำนวนข้อมูลแต่ละ genre ก่อนกำจัดข้อมูลผิดปกติ

ประเภทเพลง	จำนวนจุดข้อมูล
Underground rap	1175
Dark trap	916
Hiphop	605
Trance	599
Trap	597
Techhouse	595
Psytrance	593
Dnb	593
Techno	592
RnB	420
Trap metal	391
Rap	370
Emo	336
Hardstyle	126
Pop	92



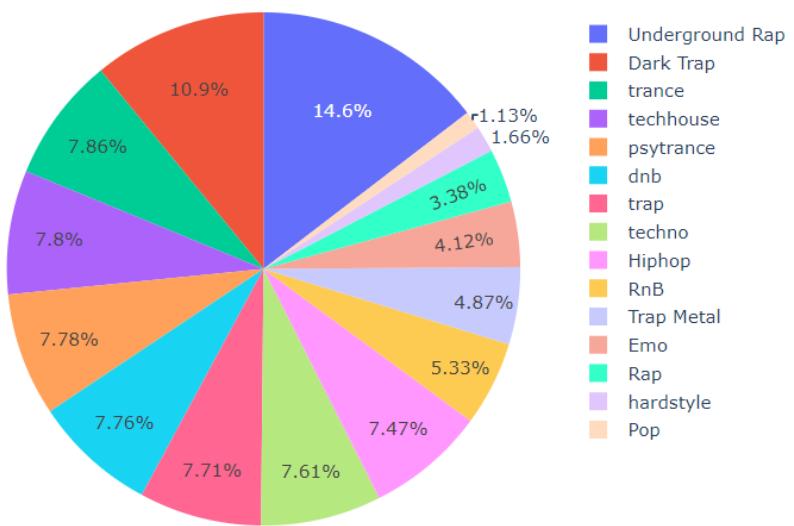
ภาพที่ 4.2.1 กราฟวงกลมแสดงจำนวนข้อมูลแต่ละ Genre ก่อนกำจัดข้อมูลผิดปกติ

จากภาพที่ 4.2.1 และตารางที่ 4.2.5 จำนวนข้อมูลในลักษณะการแบ่งของ Genre ที่มากที่สุดคือ Underground Rap มีอยู่ 1,175 ตัว คิดเป็นร้อยละ 14.7 รองลงมาเป็น Dark Trap มีอยู่ 916 ตัว คิดเป็นร้อยละ 11.5 และจำนวนข้อมูลที่น้อยที่สุดคือ Pop มีอยู่ 92 ตัว คิดเป็นร้อยละ 1.15

ตารางที่ 4.2.6 แสดงจำนวนข้อมูลแต่ละ genre หลังกำจัดข้อมูลผิดปกติ

ประเภทเพลง	จำนวนจุดข้อมูล
Underground rap	1109
Dark trap	831
Trance	597
Techhouse	593
Psytrance	591
Dnb	590
Trap	586
Techno	578
Hiphop	568
RnB	405
Trap metal	370
Emo	313
Rap	257
Hardstyle	126
Pop	86

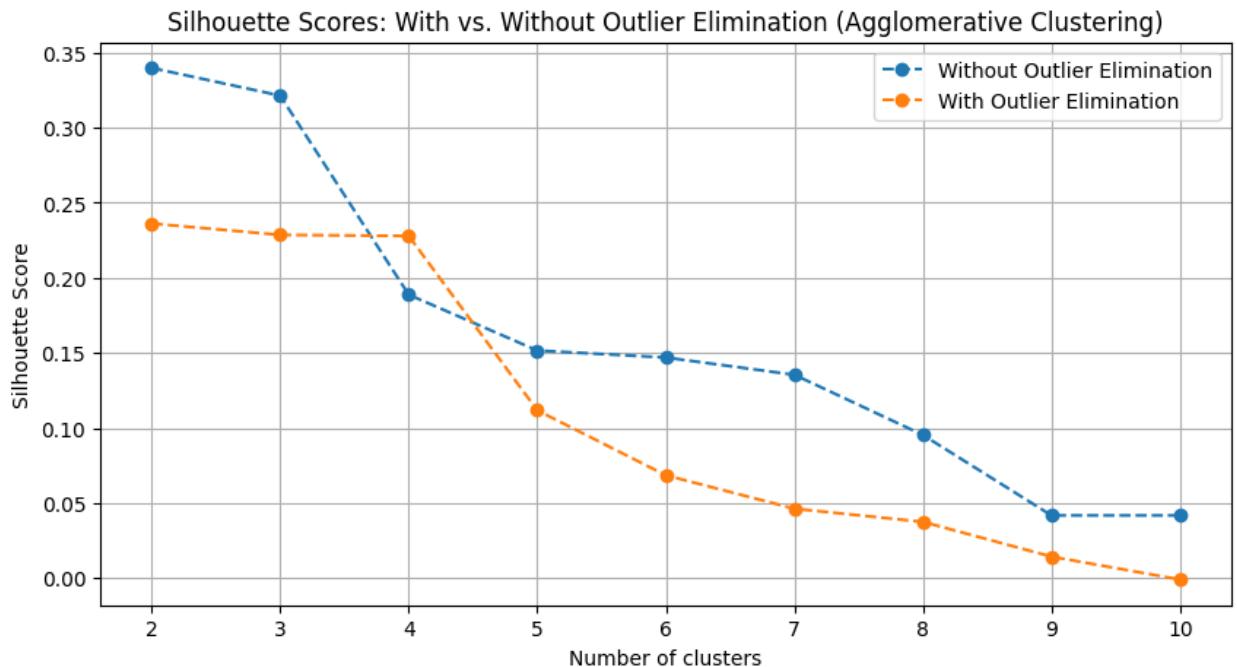
Pie Chart of Genre Counts (Cleaned Data)



ภาพที่ 4.2.2 กราฟวงกลมแสดงจำนวนข้อมูลแต่ละ Genre หลังกำจัดข้อมูลผิดปกติ

จากภาพที่ 4.2.2 และตารางที่ 4.2.6 จำนวนข้อมูลในลักษณะการแบ่งของ Genre ที่มากที่สุดคือ Underground Rap มีอยู่ 1,109 ตัว คิดเป็นร้อยละ 14.59 รองลงมาเป็น Dark Trap มีอยู่ 831 ตัว คิดเป็นร้อยละ 10.93 และจำนวนข้อมูลที่น้อยที่สุดคือ Pop มีอยู่ 86 ตัว คิดเป็นร้อยละ 1.13

#### 4.2.2 ผลการวิเคราะห์การจัดกลุ่มข้อมูลด้วยวิธี k-mean และ agglomerative single linkage

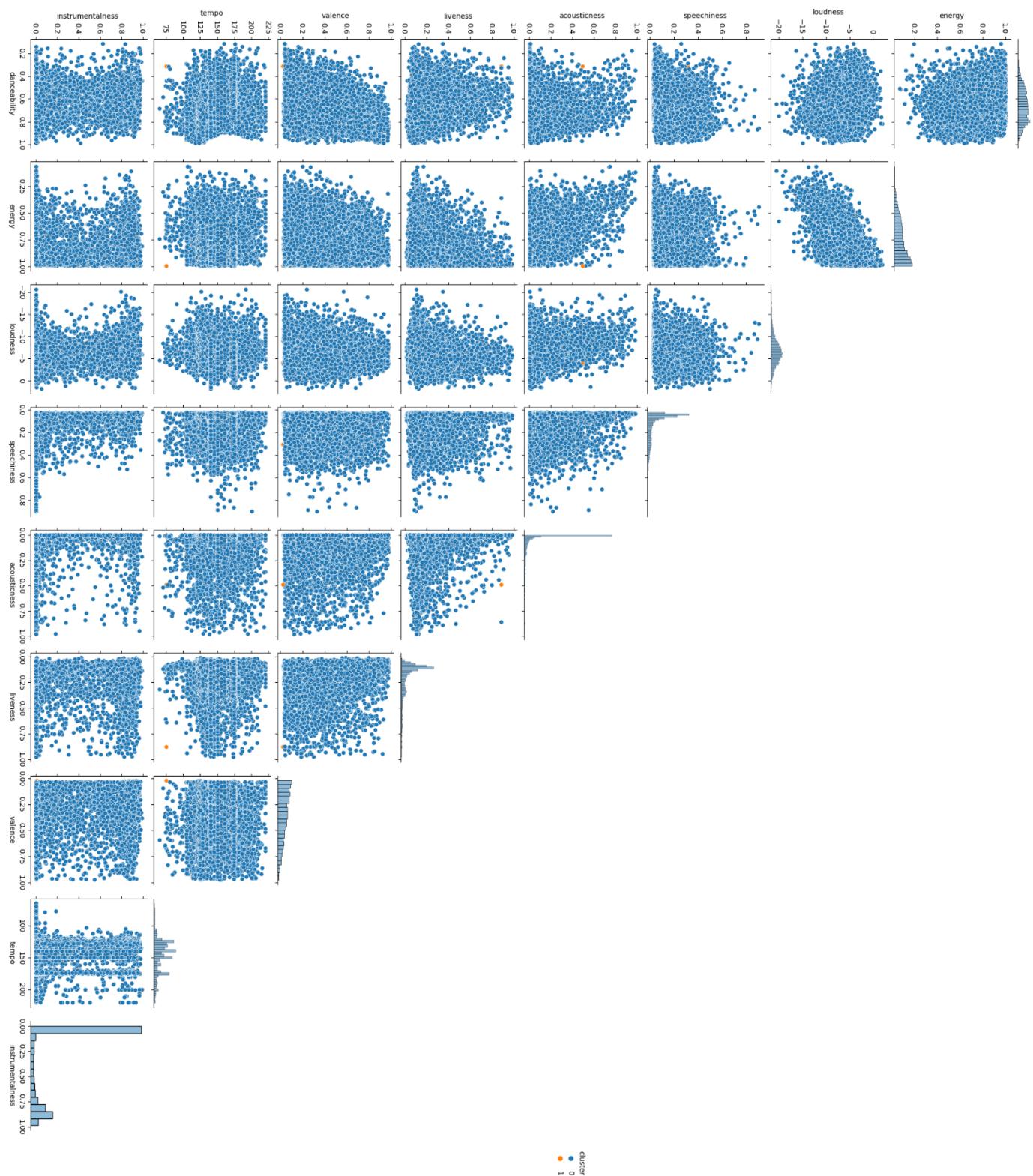


ภาพที่ 4.2.3 กราฟแสดงเปรียบเทียบค่า Silhouette ของการจัดกลุ่มโดยวิธี agglomerative single linkage ก่อนและหลังกำจัดข้อมูลผิดปกติ

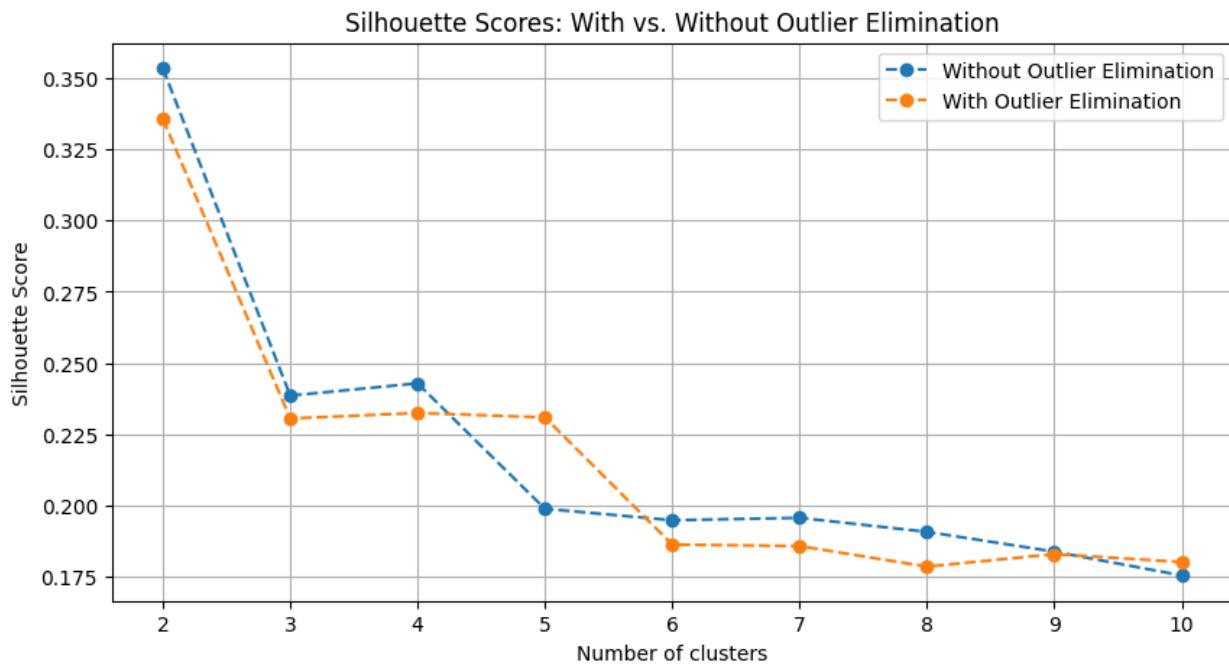
ตารางที่ 4.2.7 แสดงค่า Silhouette ของการจัดกลุ่มโดยวิธี agglomerative single linkage ก่อนและหลังกำจัดข้อมูลผิดปกติ

จำนวนกลุ่ม	Silhouette Score (Without Outlier Elimination)	Silhouette Score (With Outlier Elimination)
2	0.3398	0.2362
3	0.3214	0.2286
4	0.1888	0.2279
5	0.1516	0.1117
6	0.1469	0.0684
7	0.1354	0.0462
8	0.0953	0.0374
9	0.0417	0.0143
10	0.0417	-0.0010

จากภาพที่ 4.2.3 และตารางที่ 4.2.7 ค่า Silhouette ก่อนกำจัดข้อมูลผิดปกติที่สูงที่สุดคือ 0.3398 จำนวนกลุ่มที่จัดคือ 2 กลุ่ม ค่า Silhouette ก่อนกำจัดข้อมูลผิดปกติของลงมาคือ 0.3214 จำนวนกลุ่มที่จัดคือ 3 กลุ่ม ค่า Silhouette ก่อนกำจัดข้อมูลผิดปกติที่น้อยที่สุดคือ 0.0417 จำนวนกลุ่มที่จัดคือ 10 กลุ่ม และค่า Silhouette หลังกำจัดข้อมูลผิดปกติที่สูงที่สุดคือ 0.2362 จำนวนกลุ่มที่จัดคือ 2 กลุ่ม ค่า Silhouette หลังกำจัดข้อมูลผิดปกติ รองลงมาคือ 0.2286 จำนวนกลุ่มที่จัดคือ 3 กลุ่ม ค่า Silhouette หลังกำจัดข้อมูลผิดปกติที่น้อยที่สุดคือ -0.001 จำนวนกลุ่มที่จัดคือ 10 กลุ่ม



ภาพที่ 4.2.4 กราฟแสดงการจัดกลุ่มในแต่ละลักษณะข้อมูลโดยวิธี agglomerative single linkage ก่อนกำจัดข้อมูลผิดปกติกำหนดการแบ่งกลุ่ม 2 กลุ่ม

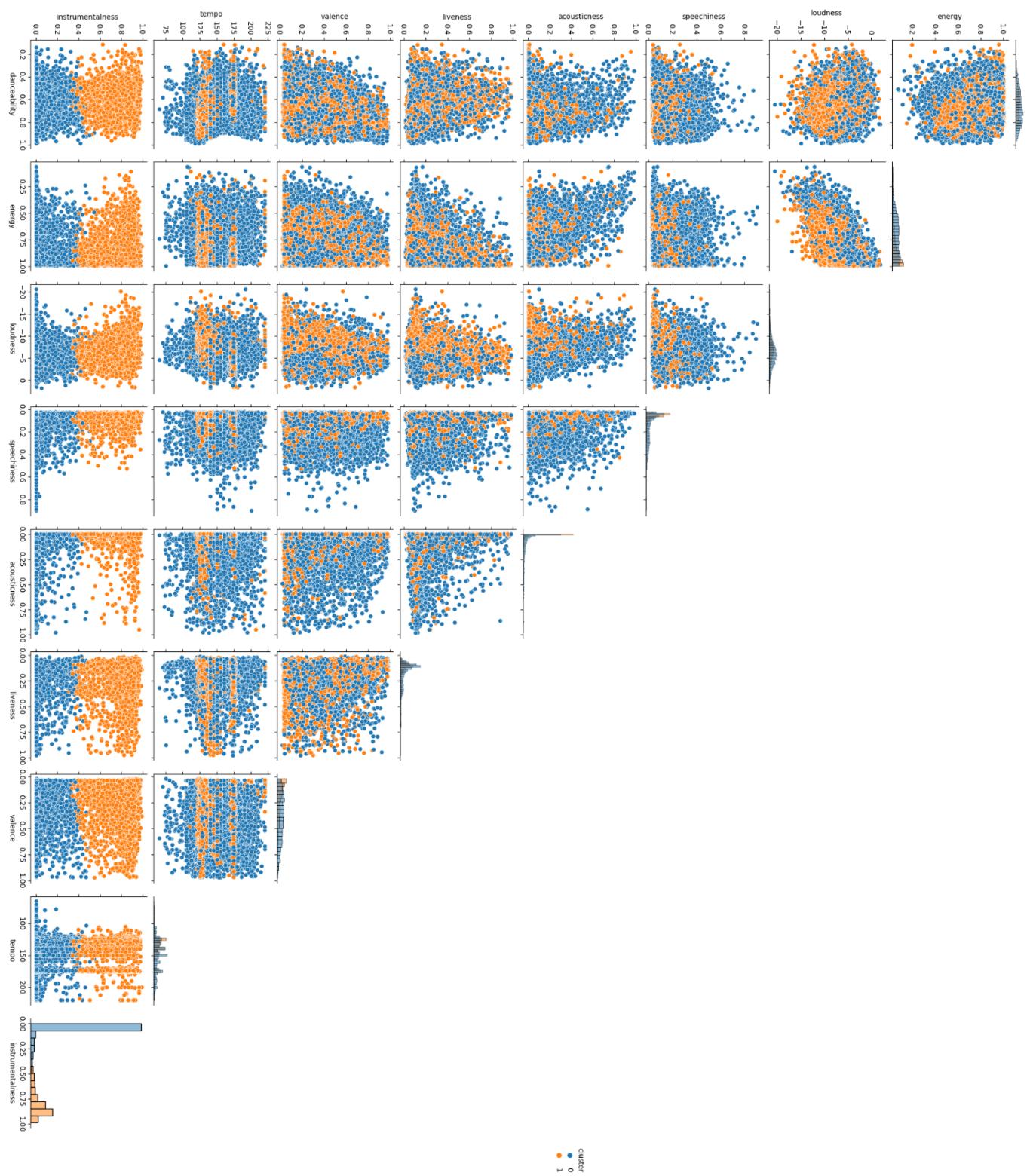


ภาพที่ 4.2.5 กราฟเปรียบค่า silhouette การจัดกลุ่มด้วยวิธี k-mean ระหว่างก่อนและหลังกำจัดข้อมูลผิดปกติ

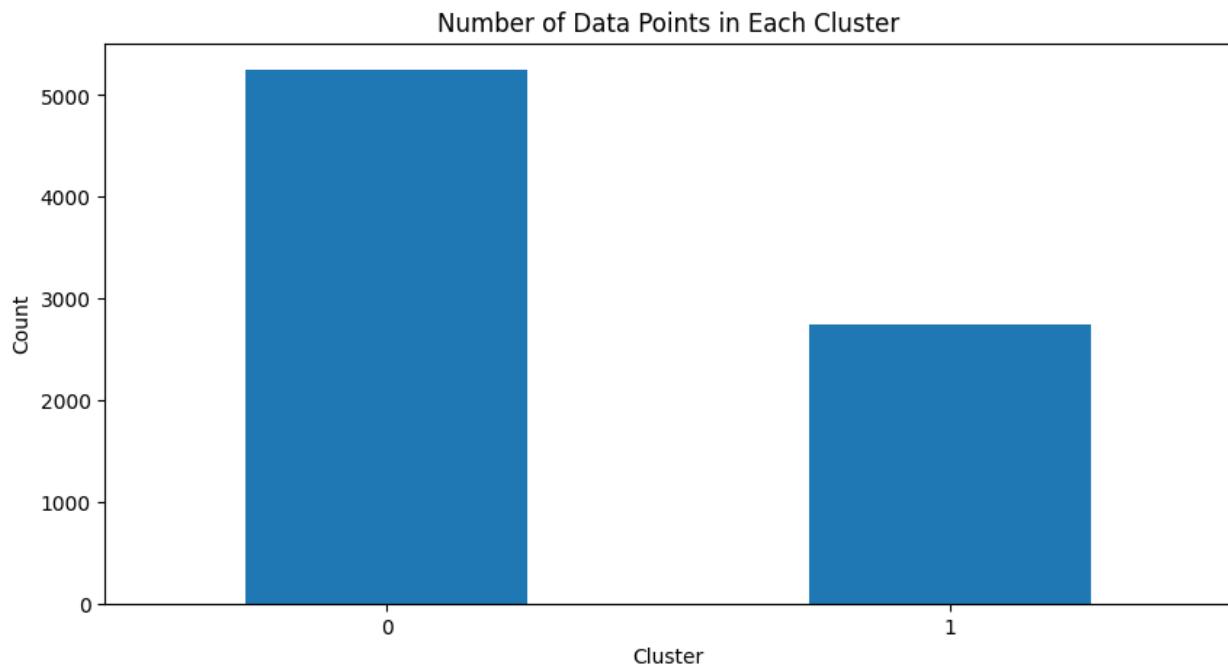
ตารางที่ 4.2.8 แสดงค่า Silhouette ของการจัดกลุ่มโดยวิธี k-mean ก่อนและหลังกำจัดข้อมูลผิดปกติ

Number of Cluster	Silhouette Score (Without Outlier Elimination)	Silhouette Score (With Outlier Elimination)
2	0.3534	0.3356
3	0.2385	0.2305
4	0.2428	0.2324
5	0.1988	0.2309
6	0.1948	0.1862
7	0.1956	0.1857
8	0.1907	0.1785
9	0.1838	0.1829
10	0.1754	0.1801

จากภาพที่ 4.2.5 และตารางที่ 4.2.8 ค่า Silhouette ก่อนกำจัดข้อมูลผิดปกติที่สูงที่สุดคือ 0.3534 จำนวนกลุ่มที่จัดคือ 2 กลุ่ม ค่า Silhouette ก่อนกำจัดข้อมูลผิดปกติของลงมาคือ 0.2428 จำนวนกลุ่มที่จัดคือ 4 กลุ่ม ค่า Silhouette ก่อนกำจัดข้อมูลผิดปกติที่น้อยที่สุดคือ 0.1754 จำนวนกลุ่มที่จัดคือ 10 กลุ่ม และค่า Silhouette หลังกำจัดข้อมูลผิดปกติที่สูงที่สุดคือ 0.3356 จำนวนกลุ่มที่จัดคือ 2 กลุ่ม ค่า Silhouette หลังกำจัดข้อมูลผิดปกติของลงมาคือ 0.2324 จำนวนกลุ่มที่จัดคือ 4 กลุ่ม ค่า Silhouette หลังกำจัดข้อมูลผิดปกติที่น้อยที่สุดคือ 0.1785 จำนวนกลุ่มที่จัดคือ 8 กลุ่ม



ภาพที่ 4.2.6 กราฟแสดงการจัดกลุ่มในแต่ละลักษณะข้อมูลโดยวิธี k-mean ก่อนกำจัดข้อมูล  
ผิดปกติกำหนดการแบ่งกลุ่ม 2 กลุ่ม

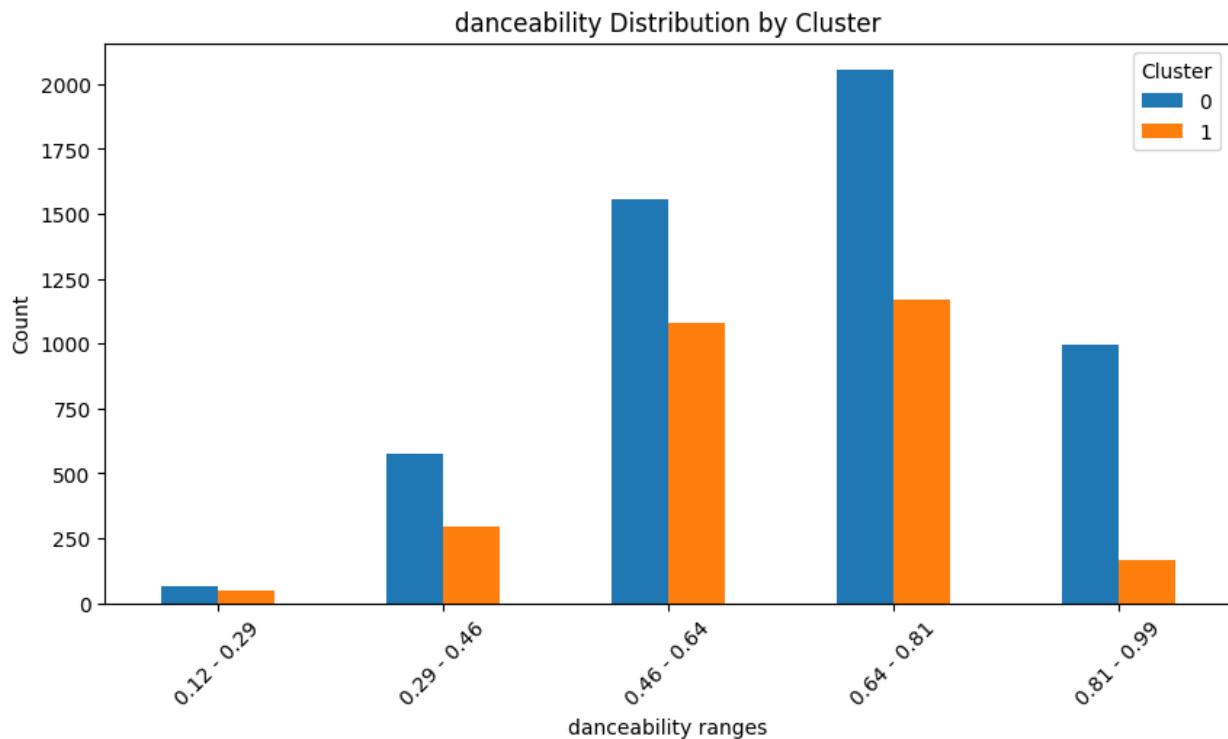


ภาพที่ 4.2.7 แสดงจำนวนข้อมูลแต่ละกลุ่มโดยวิธี k-mean

ตารางที่ 4.2.9 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean ของจำนวนข้อมูลแต่ละกลุ่ม

กลุ่ม	จำนวนจุดข้อมูล
0	5249
1	2751

จากภาพที่ 4.2.7 และตารางที่ 4.2.9 จำนวนกลุ่มที่มากที่สุดคือ กลุ่ม 0 มีจำนวน 5,249 ตัว คิดเป็นร้อยละ 65.61 จำนวนกลุ่มรองลงมาคือ กลุ่ม 1 มีจำนวน 2,751 ตัว คิดเป็นร้อยละ 34.39

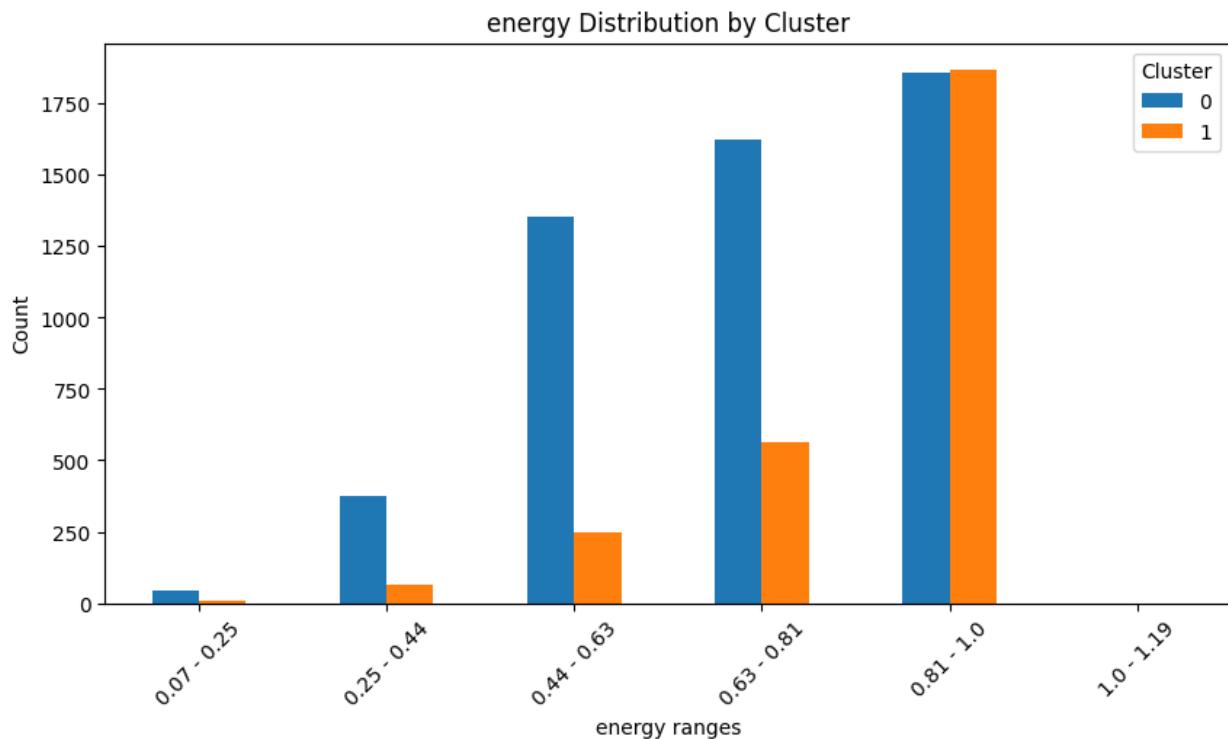


ภาพที่ 4.2.8 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean สำหรับลักษณะ danceability

ตารางที่ 4.2.10 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยลักษณะ danceability แบ่งเป็น 5 ช่วงย่อย

ช่วง Danceability	กลุ่ม 0	กลุ่ม 1
0.12 – 0.29	62	45
0.29 – 0.46	573	295
0.46 – 0.64	1558	1078
0.64 – 0.81	2056	1170
0.81 – 1.00	998	163

จากภาพที่ 4.2.8 และตารางที่ 4.2.10 ข้อมูลลักษณะ danceability ของกลุ่ม 0 ส่วนใหญ่อยู่ในช่วง 0.64 – 0.81 คิดเป็นร้อยละ 39.17 จากจำนวนกลุ่ม 0 ทั้งหมด รองลงมาอยู่ในช่วง 0.46 – 0.64 คิดเป็นร้อยละ 29.68 จากจำนวนกลุ่ม 0 ทั้งหมด และข้อมูลลักษณะ danceability ของกลุ่ม 1 ส่วนใหญ่อยู่ในช่วง 0.64 – 0.81 คิดเป็นร้อยละ 42.53 จากจำนวนกลุ่ม 1 ทั้งหมด รองลงมาอยู่ในช่วง 0.46 – 0.64 คิดเป็นร้อยละ 39.18 จากจำนวนกลุ่ม 1 ทั้งหมด

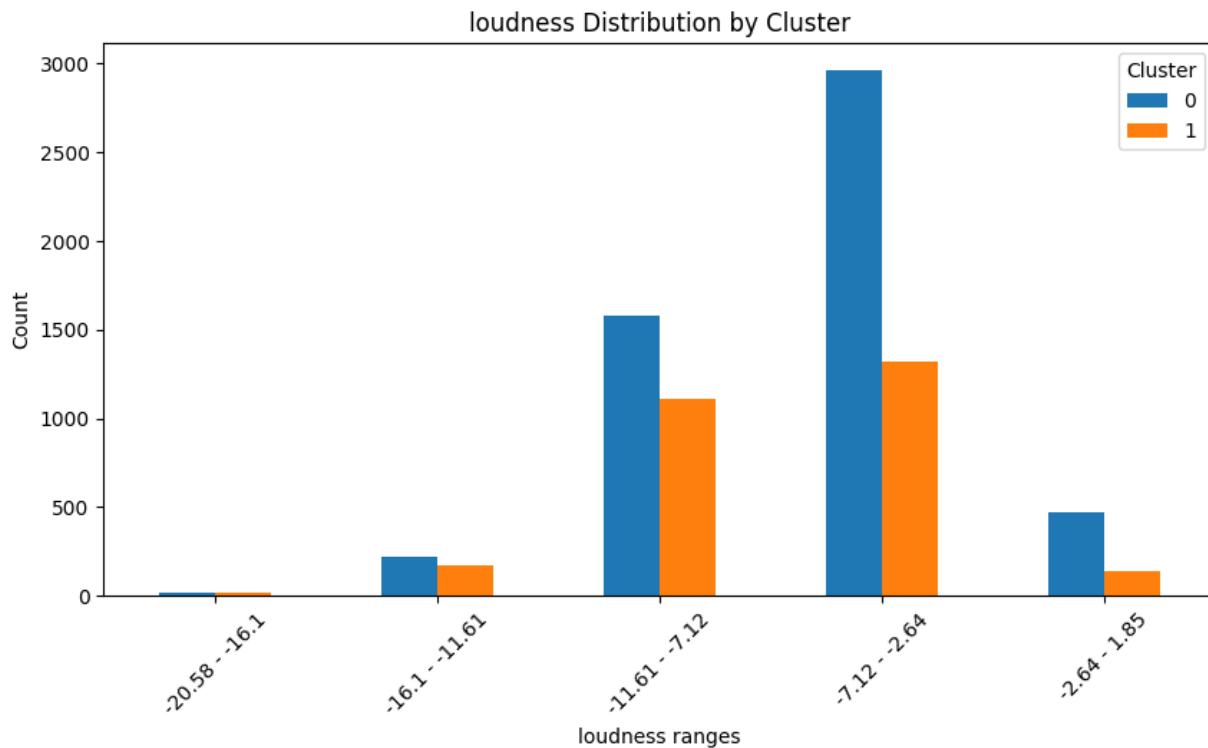


ภาพที่ 4.2.9 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean สำหรับลักษณะ energy

ตารางที่ 4.2.11 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยลักษณะ energy แบ่งเป็น 5 ช่วงย่อย

ช่วง Energy	กลุ่ม 0	กลุ่ม 1
0.07 – 0.25	42	10
0.25 – 0.44	374	62
0.44 – 0.63	1353	248
0.63 – 0.81	1623	563
0.81 – 1.00	1857	1868

จากภาพที่ 4.2.9 และตารางที่ 4.2.11 ข้อมูลลักษณะ energy ของกลุ่ม 0 ส่วนใหญ่อยู่ในช่วง 0.81 – 1.00 คิดเป็นร้อยละ 35.38 จากจำนวนกลุ่ม 0 ทั้งหมด รองลงมาอยู่ในช่วง 0.63 – 0.81 คิดเป็นร้อยละ 30.92 จากจำนวนกลุ่ม 0 ทั้งหมด และข้อมูลลักษณะ energy ของกลุ่ม 1 ส่วนใหญ่อยู่ในช่วง 0.81 – 1.00 คิดเป็นร้อยละ 67.90 จากจำนวนกลุ่ม 1 ทั้งหมด รองลงมาอยู่ในช่วง 0.63 – 0.81 คิดเป็นร้อยละ 20.46 จากจำนวนกลุ่ม 1 ทั้งหมด

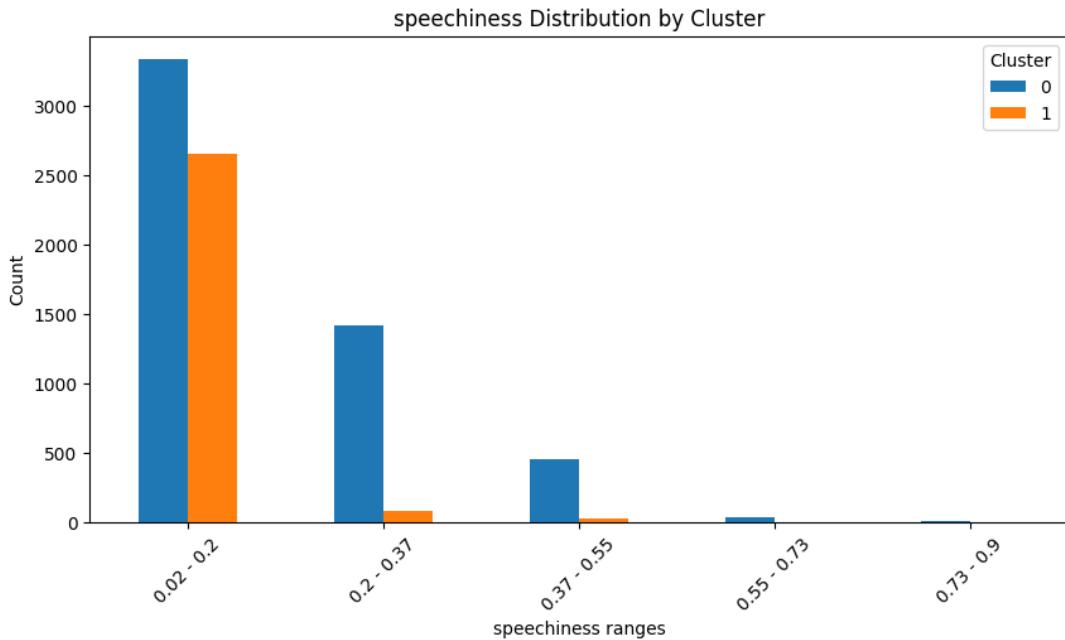


ภาพที่ 4.2.10 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean สำหรับลักษณะ loudness

ตารางที่ 4.2.12 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยลักษณะ loudness แบ่งเป็น 5 ช่วงย่อย

ช่วง Loudness (dB)	กลุ่ม 0	กลุ่ม 1
-20.58 - -16.1	16	18
-16.1 - -11.61	220	168
-11.61 - -7.12	1577	1113
-7.12 - -2.64	2967	1317
-2.64 - 1.85	469	135

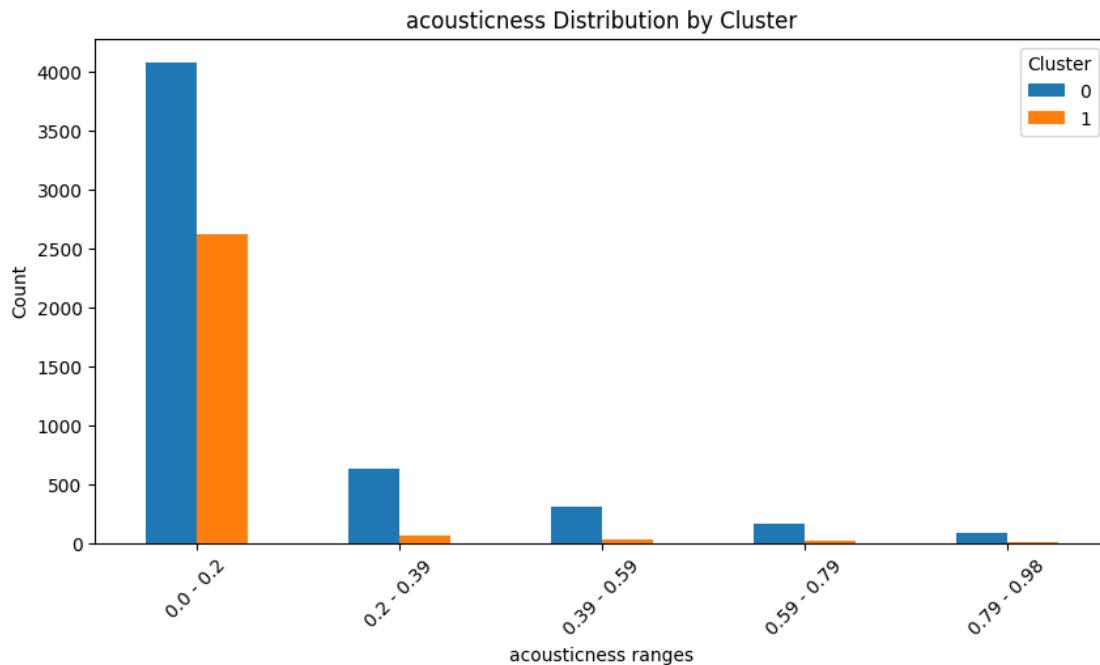
จากภาพที่ 4.2.10 และตารางที่ 4.2.12 ข้อมูลลักษณะ loudness ของกลุ่ม 0 ส่วนใหญ่อยู่ในช่วง -7.12 ถึง -2.64 dB คิดเป็นร้อยละ 54.65 จากจำนวนกลุ่ม 0 ทั้งหมด รองลงอยู่ในช่วง -11.61 ถึง -7.12 dB คิดเป็นร้อยละ 30.04 จากจำนวนกลุ่ม 0 ทั้งหมด และข้อมูลลักษณะ loudness ของกลุ่ม 1 ส่วนใหญ่อยู่ในช่วง -7.12 ถึง -2.64 dB คิดเป็นร้อยละ 47.87 จากจำนวนกลุ่ม 1 ทั้งหมด รองลงมาอยู่ในช่วง -11.61 ถึง -7.12 dB คิดเป็นร้อยละ 40.46 จากจำนวนกลุ่ม 1 ทั้งหมด



ภาพที่ 4.2.11 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean สำหรับลักษณะ speechiness ตารางที่ 4.2.13 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยลักษณะ speechiness แบ่งเป็น 5 ช่วงย่อย

ช่วง Speechiness	กลุ่ม 0	กลุ่ม 1
0.02 - 0.2	3332	2648
0.2 - 0.37	1420	80
0.37 - 0.55	452	23
0.55 - 0.73	34	0
0.73 - 0.9	11	0

จากภาพที่ 4.2.11 และตารางที่ 4.2.13 ข้อมูลลักษณะ speechiness ของกลุ่ม 0 ส่วนใหญ่อยู่ในช่วง 0.02 - 0.2 คิดเป็นร้อยละ 63.48 จากจำนวนกลุ่ม 0 ทั้งหมด รองลงอยู่ในช่วง 0.2 - 0.37 คิดเป็นร้อยละ 27.05 จากจำนวนกลุ่ม 0 ทั้งหมด และข้อมูลลักษณะ speechiness ของกลุ่ม 1 ส่วนใหญ่อยู่ในช่วง 0.02 - 0.2 คิดเป็นร้อยละ 96.25 จากจำนวนกลุ่ม 1 ทั้งหมด รองลงมาอยู่ในช่วง 0.2 - 0.37 คิดเป็นร้อยละ 2.91 จากจำนวนกลุ่ม 1 ทั้งหมด

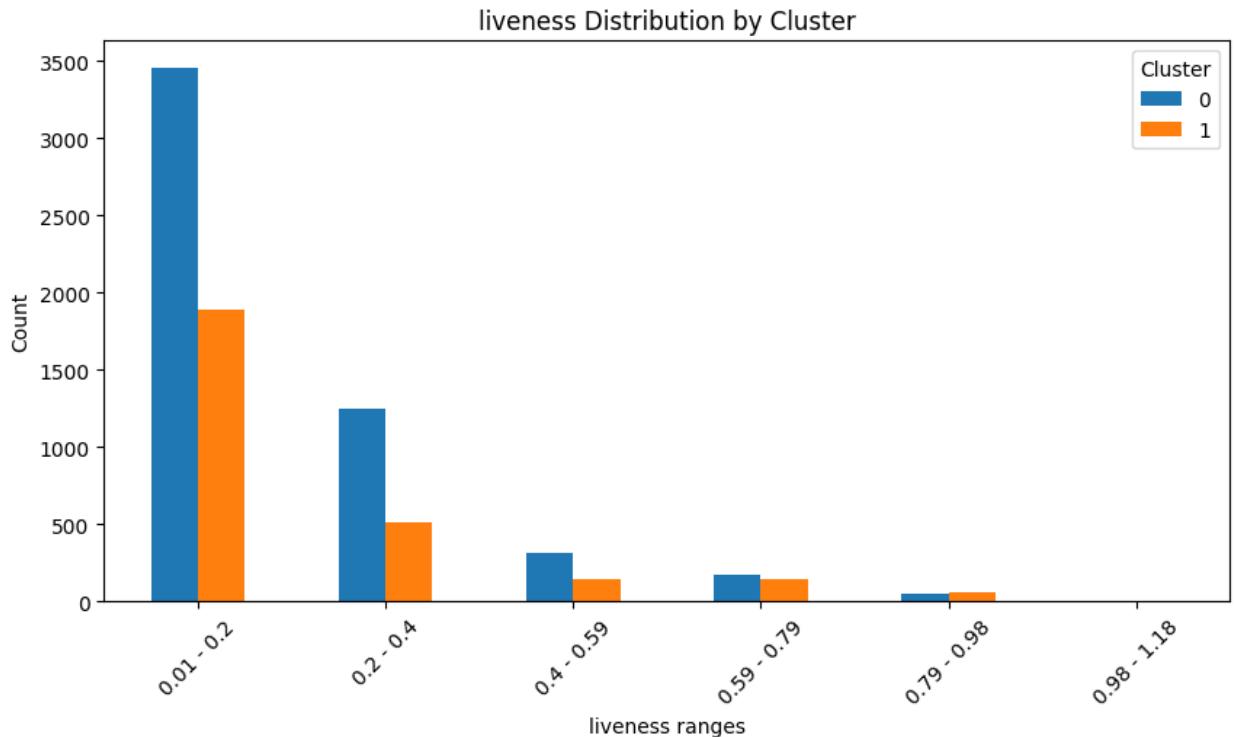


ภาพที่ 4.2.12 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean สำหรับลักษณะ acousticness

ตารางที่ 4.2.14 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยลักษณะ acousticness แบ่งเป็น 5 ช่วงย่อย

ช่วง Acousticness	กลุ่ม 0	กลุ่ม 1
0.0 - 0.2	4074	2622
0.2 - 0.39	629	65
0.39 - 0.59	302	33
0.59 - 0.79	164	20
0.79 - 0.98	80	11

จากภาพที่ 4.2.12 และตารางที่ 4.2.14 ข้อมูลลักษณะ acousticness ของกลุ่มที่ 0 ส่วนใหญ่อยู่ในช่วง 0.0 - 0.2 คิดเป็นร้อยละ 77.61 จากจำนวนกลุ่มที่ 0 ทั้งหมด รองลงมาอยู่ในช่วง 0.2 - 0.39 คิดเป็นร้อยละ 11.98 จากจำนวนกลุ่มที่ 0 ทั้งหมด และข้อมูลลักษณะ acousticness ของกลุ่มที่ 1 ส่วนใหญ่อยู่ในช่วง 0.0 - 0.2 คิดเป็นร้อยละ 95.31 จากจำนวนกลุ่มที่ 1 ทั้งหมด รองลงมาอยู่ในช่วง 0.2 - 0.39 คิดเป็นร้อยละ 2.36 จากจำนวนกลุ่มที่ 1 ทั้งหมด

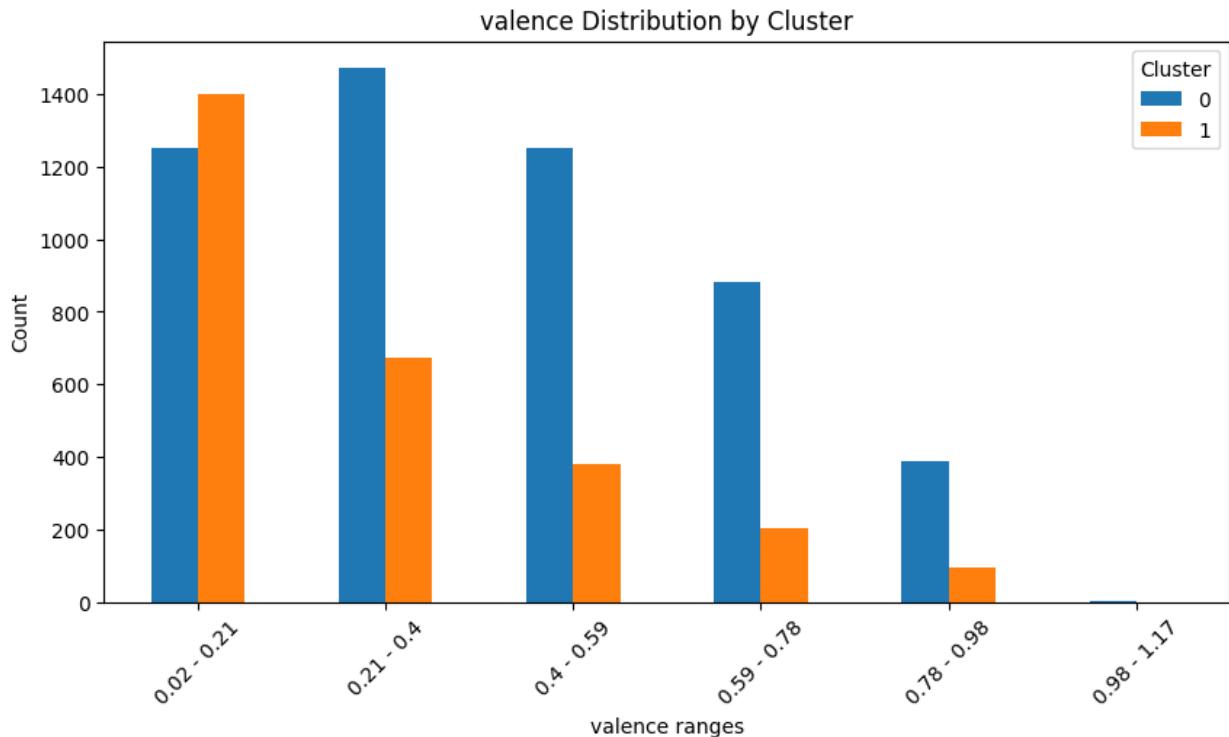


ภาพที่ 4.2.13 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean สำหรับลักษณะ liveness

ตารางที่ 4.2.15 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยลักษณะ liveness แบ่งเป็น 5 ช่วงย่อย

ช่วง Liveness	กลุ่ม 0	กลุ่ม 1
0.01 - 0.2	3458	1887
0.2 - 0.4	1251	513
0.4 - 0.59	315	150
0.59 - 0.79	170	143
0.79 - 0.98	55	58

จากภาพที่ 4.2.13 และตารางที่ 4.2.15 ข้อมูลลักษณะ liveness ของกลุ่มที่ 0 ส่วนใหญ่อยู่ในช่วง 0.01 - 0.2 คิดเป็นร้อยละ 65.88 จากจำนวนกลุ่มที่ 0 ทั้งหมด รองลงมาอยู่ในช่วง 0.2 - 0.4 คิดเป็นร้อยละ 23.83 จากจำนวนกลุ่มที่ 0 ทั้งหมด และข้อมูลลักษณะ liveness ของกลุ่มที่ 1 ส่วนใหญ่อยู่ในช่วง 0.01 - 0.2 คิดเป็นร้อยละ 68.59 จากจำนวนกลุ่ม 1 ทั้งหมด รองลงมาอยู่ในช่วง 0.2 - 0.4 คิดเป็นร้อยละ 18.65 จากจำนวนกลุ่มที่ 1 ทั้งหมด

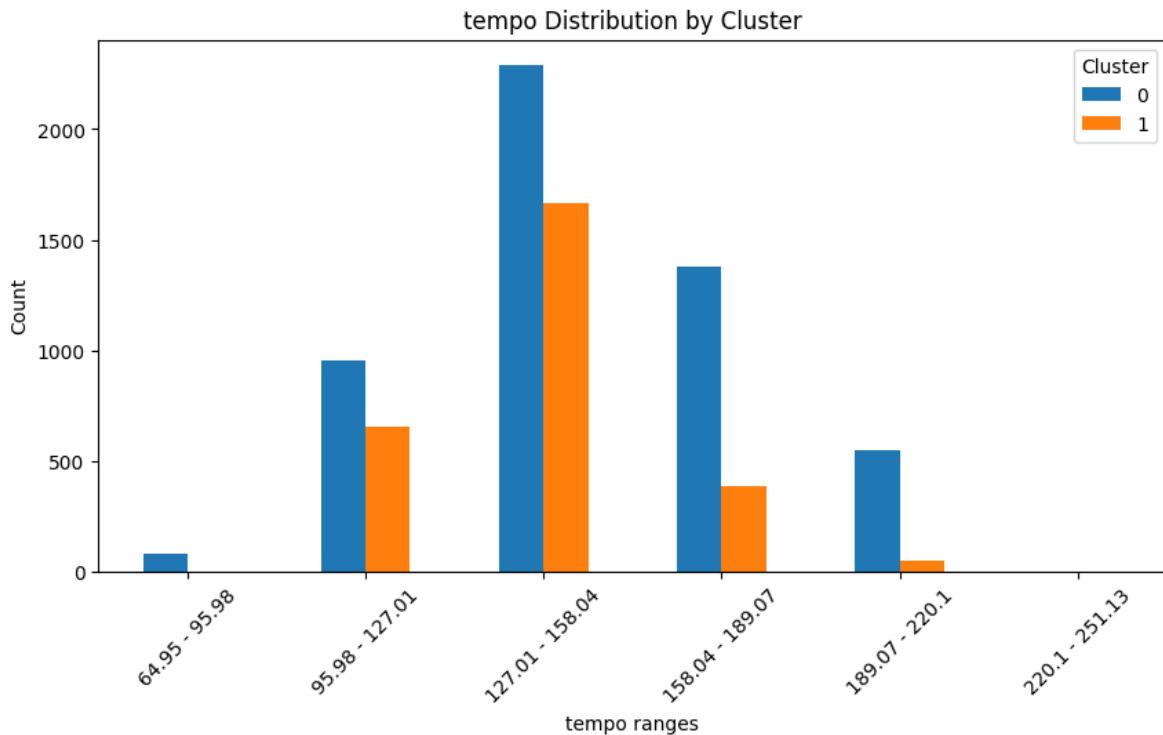


ภาพที่ 4.2.14 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean สำหรับลักษณะ valence

ตารางที่ 4.2.16 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยลักษณะ valence แบ่งเป็น 5 ช่วงย่อย

Valence Bin	กลุ่ม 0	กลุ่ม 1
0.02 - 0.21	1251	1402
0.21 - 0.4	1473	673
0.4 - 0.59	1252	379
0.59 - 0.79	883	204
0.79 - 0.98	390	93

จากภาพที่ 4.2.14 และตารางที่ 4.2.16 ข้อมูลลักษณะ valence ของกลุ่ม 0 ส่วนใหญ่อยู่ในช่วง 0.21 - 0.4 คิดเป็นร้อยละ 28.06 จากจำนวนกลุ่ม 0 ทั้งหมด รองลงมาอยู่ในช่วง 0.4 - 0.59 คิดเป็นร้อยละ 23.85 จากจำนวนกลุ่ม 0 ทั้งหมด และข้อมูลลักษณะ valence ของกลุ่ม 1 ส่วนใหญ่อยู่ในช่วง 0.02 - 0.21 คิดเป็นร้อยละ 50.96 จากจำนวนกลุ่ม 1 ทั้งหมด รองลงมาอยู่ในช่วง 0.21 - 0.4 คิดเป็นร้อยละ 24.46 จากจำนวนกลุ่ม 1 ทั้งหมด

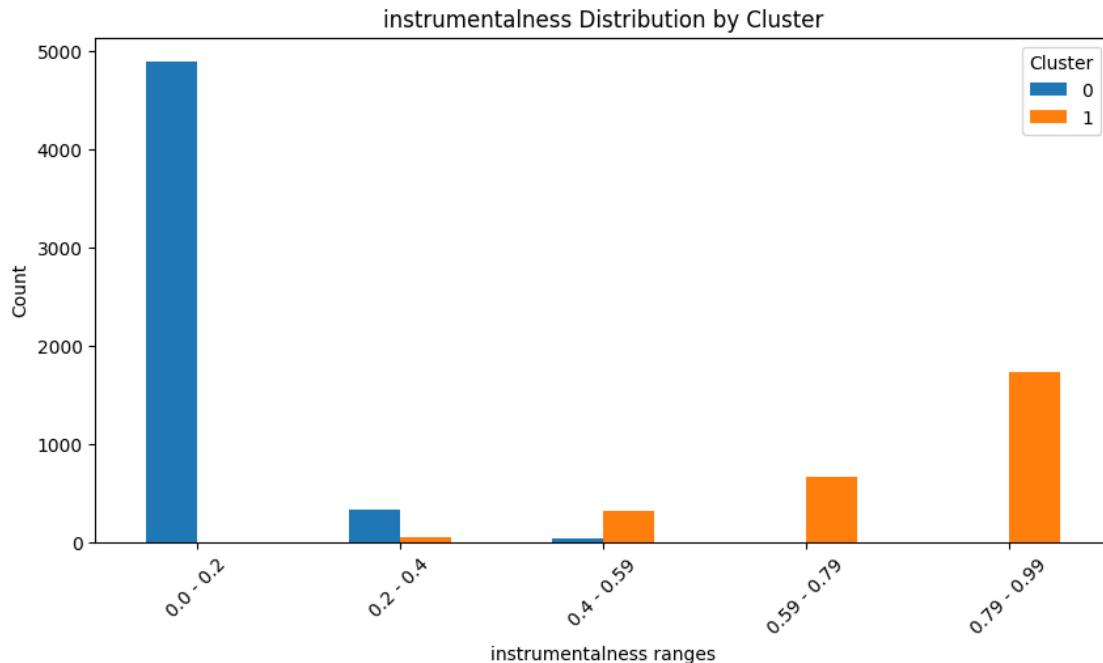


ภาพที่ 4.2.15 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean สำหรับลักษณะ tempo

ตารางที่ 4.2.17 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยลักษณะ tempo แบ่งเป็น 5 ช่วงย่อย

ช่วง Tempo (BPM)	กลุ่ม 0	กลุ่ม 1
64.95 - 95.98	80	0
95.98 - 127.01	955	653
127.01 - 158.04	2290	1666
158.04 - 189.07	1378	387
189.07 - 220.1	546	45

จากภาพที่ 4.2.15 และตารางที่ 4.2.17 ข้อมูลลักษณะ tempo ของกลุ่ม 0 ส่วนใหญ่อยู่ในช่วง 127.01 - 158.04 BPM คิดเป็นร้อยละ 43.63 จากจำนวนกลุ่ม 0 ทั้งหมด รองลงมาอยู่ในช่วง 158.04 - 189.07 BPM คิดเป็นร้อยละ 26.25 จากจำนวนกลุ่ม 0 ทั้งหมด และข้อมูลลักษณะ tempo ของกลุ่ม 1 ส่วนใหญ่อยู่ในช่วง 127.01 - 158.04 BPM คิดเป็นร้อยละ 60.56 จากจำนวนกลุ่ม 1 ทั้งหมด รองลงมาอยู่ในช่วง 95.98 - 127.01 BPM คิดเป็นร้อยละ 23.74 จากจำนวนกลุ่ม 1 ทั้งหมด

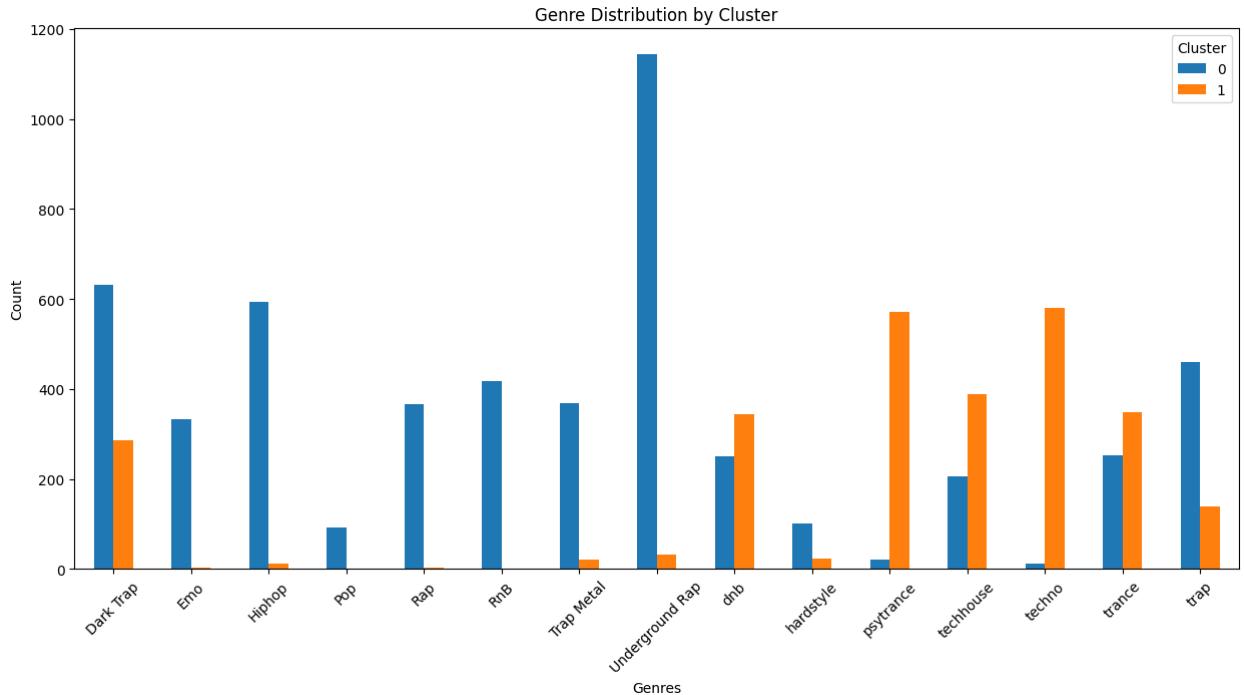


ภาพที่ 4.2.16 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean สำหรับลักษณะ instrumentalness

ตารางที่ 4.2.18 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยลักษณะ instrumentalness แบ่งเป็น 5 ช่วงโดย

ช่วง Instrumentalness	กลุ่ม 0	กลุ่ม 1
0.0 - 0.2	4891	0
0.2 - 0.4	324	42
0.4 - 0.59	34	316
0.59 - 0.79	0	664
0.79 - 0.99	0	1729

จากภาพที่ 4.2.16 และตารางที่ 4.2.18 ข้อมูลลักษณะ instrumentalness ของกลุ่ม 0 ส่วนใหญ่อยู่ในช่วง 0.0 - 0.2 คิดเป็นร้อยละ 93.18 จากจำนวนกลุ่ม 0 ทั้งหมด รองลงมาอยู่ในช่วง 0.2 - 0.4 คิดเป็นร้อยละ 6.17 จากจำนวนกลุ่ม 0 ทั้งหมด และข้อมูลลักษณะ instrumentalness ของกลุ่ม 1 ส่วนใหญ่อยู่ในช่วง 0.79 - 0.99 คิดเป็นร้อยละ 62.85 จากจำนวนกลุ่ม 1 ทั้งหมด รองลงมาอยู่ในช่วง 0.59 - 0.79 คิดเป็นร้อยละ 24.14 จากจำนวนกลุ่ม 1 ทั้งหมด



ภาพที่ 4.2.17 แสดงผลลัพธ์การจัดกลุ่มข้อมูลโดยวิธี k-mean แบ่งตามประเภทของเพลง

ตารางที่ 4.2.19 แสดงจำนวนข้อมูลกลุ่มที่ 0 โดยวิธี k-mean แบ่งตามประเภทของเพลง

ประเภทของเพลง	จำนวนข้อมูล
Underground Rap	1144
Dark Trap	631
Hiphop	594
trap	459
RnB	418
Trap Metal	369
Rap	367
Emo	332
trance	252
dnb	250
techhouse	206
hardstyle	102
Pop	93

psytrance	21
techno	11

ตารางที่ 4.2.20 แสดงจำนวนข้อมูลกลุ่มที่ 1 โดยวิธี k-mean แบ่งตามประเภทของเพลง

ประเภทของเพลง	จำนวนข้อมูล
techno	580
psytrance	571
techhouse	389
trance	348
dnb	343
Dark Trap	285
trap	139
Underground Rap	31
hardstyle	24
Trap Metal	22
Hiphop	11
Emo	4
Rap	3
RnB	3
Pop	0

จากภาพที่ 4.2.17 ตารางที่ 4.2.19 และตารางที่ 4.2.20 จำนวนข้อมูล 5 อันดับแรก ในลักษณะประเภท เพลงของกลุ่มที่ 0 รวมคิดเป็นร้อยละ 61.84 จากข้อมูลกลุ่ม 0 ทั้งหมด แบ่งเป็นประเภทเพลง Underground Rap (21.79%), Dark Trap (12.02%), Hiphop (11.32%), trap (8.74%) และ RnB (7.96%) และจำนวนข้อมูล 5 อันดับแรก ในลักษณะประเภทเพลงของกลุ่มที่ 1 รวมคิดเป็นร้อยละ 81.10 จากข้อมูลกลุ่ม 1 ทั้งหมด แบ่งเป็น ประเภทเพลง techno (21.08%), psytrance (21.79%), techhouse (14.14%), trance (12.65%) และ dnb (12.47%)

## บทที่ 5

### ข้อสรุปและข้อเสนอแนะ

หลังจากผู้จัดทำได้ผลลัพธ์ต่างๆ ออกมาจึงได้นำมาวิเคราะห์ผลลัพธ์เหล่านี้ โดยการตรวจจับข้อมูล ผิดปกติผ่านทางอัลกอริทึม MOF และนำมามีช่วงร่วมกับอัลกอริทึมการจัดกลุ่มข้อมูลได้แก่ อัลกอริทึม k-mean และอัลกอริทึม agglomerative single linkage เพื่อดูผลลัพธ์การจัดกลุ่มแบบกำจัดข้อมูลผิดปกติออกและการจัดกลุ่มแบบไม่กำจัดข้อมูลผิดปกติออก โครงการนี้จะทำการวิเคราะห์การจัดกลุ่มข้อมูลสังเคราะห์ 5 รูป่าง ได้แก่ ได้แก่ (1) วงกลมซ้อนสองวง (2) รูปพระจันทร์ซ้อนสองวง (3) กลุ่มข้อมูลกระจายสามกลุ่ม (4) กลุ่มข้อมูลที่เรียงเป็นเส้นสามเส้น (5) กลุ่มข้อมูลที่กระจายเป็นกลุ่มใหญ่ แต่ละรูป่างมีขนาดข้อมูล 400 ตัว และทดสอบการจัดกลุ่มกับข้อมูล spotify จำนวน 8,000 ตัว ซึ่งมาจากการสุ่มแบบ systematic sampling จากข้อมูลทั้งหมด 42,306 ตัว โดยผู้จัดทำมีวิธีการประเมินการจัดกลุ่มหลักๆ ด้วยกันดังนี้ การวัดค่า Silhouette การจัดกลุ่มเปรียบเทียบแบบกำจัดและไม่กำจัดข้อมูลผิดปกติ การแสดงผลทางภาพและภาพเพื่อให้เห็นถึงการจัดกลุ่มได้ชัดเจนมากยิ่งขึ้น นอกจากนี้ผู้จัดทำได้ทำการตีความการจัดกลุ่มของข้อมูล spotify ผ่านทางข้อมูลเชิงลึกในแต่ละกลุ่ม ผู้จัดทำมีข้อเสนอแนะและผลการวิเคราะห์ออกมาดังต่อไปนี้

#### 5.1 สรุปผลการวิเคราะห์

จากการวิเคราะห์ข้อมูลสังเคราะห์ 5 รูปแบบและข้อมูล spotify จำนวน 8,000 ตัว โดยการสรุปผลจะมีดังต่อไปนี้ การจัดกลุ่มข้อมูลและตรวจจับข้อมูลผิดปกติกับข้อมูลสังเคราะห์ การจัดกลุ่มข้อมูลและตรวจจับข้อมูลผิดปกติกับข้อมูล spotify และการตีความการจัดกลุ่มข้อมูล spotify

##### 5.1.1 การจัดกลุ่มข้อมูลและตรวจจับข้อมูลผิดปกติกับข้อมูลสังเคราะห์

1) ข้อมูลสังเคราะห์รูป่างวงกลมซ้อนสองวง การตรวจจับข้อมูลผิดปกติตัวอย่างอัลกอริทึม MOF ไม่ได้ส่งผลต่อการจัดกลุ่มด้วยวิธี k-mean และ agglomerative single linkage มากนักโดยค่า Silhouette ของการจัดกลุ่มด้วยวิธี k-mean หลังการกำจัดข้อมูลผิดปกติมีค่าเพิ่มขึ้นเพียง 0.0034 เพิ่มขึ้นคิดเป็นร้อยละ 0.96 ค่า Silhouette ของการจัดกลุ่มด้วยวิธี agglomerative single linkage หลังการกำจัดข้อมูลผิดปกติมีค่าเพิ่มขึ้นเพียง 0.2342 เพิ่มขึ้นคิดเป็นร้อยละ 8.73

2) ข้อมูลสังเคราะห์รูปพระจันทร์ซ้อนสองวง การตรวจจับข้อมูลผิดปกติตัวอย่างอัลกอริทึม MOF ไม่ได้ส่งผลต่อการจัดกลุ่มด้วยวิธี k-mean และ agglomerative single linkage มากนักโดยค่า Silhouette ของ

การจัดกลุ่มด้วยวิธี k-mean หลังการจำจัดข้อมูลผิดปกติมีค่าเพิ่มขึ้นเพียง 0.0023 เพิ่มขึ้นคิดเป็นร้อยละ 0.48 ค่า Silhouette ของการจัดกลุ่มด้วยวิธี agglomerative single linkage หลังการจำจัดข้อมูลผิดปกติมีค่าเพิ่มขึ้นเพียง 0.0049 เพิ่มขึ้นคิดเป็นร้อยละ 1.48

3) ข้อมูลสังเคราะห์รูปร่างกลุ่มข้อมูลกระจายสามกลุ่ม การตรวจจับข้อมูลผิดปกติด้วยอัลกอริทึม MOF ไม่ได้ส่งผลต่อการจัดกลุ่มด้วยวิธี k-mean มากนักโดยค่า Silhouette ของการจัดกลุ่มด้วยวิธี k-mean หลังการจำจัดข้อมูลผิดปกติมีค่าเพิ่มขึ้นเพียง 0.0182 เพิ่มขึ้นคิดเป็นร้อยละ 2.33 แต่ในทางกลับกันอัลกอริทึมการตรวจจับข้อมูลผิดปกติ MOF ส่งผลต่อการจัดกลุ่มด้วยวิธี agglomerative single linkage อย่างเห็นได้ชัด ค่า Silhouette ของการจัดกลุ่มด้วยวิธี agglomerative single linkage หลังการจำจัดข้อมูลผิดปกติมีค่าเพิ่มขึ้น 0.4466 เพิ่มขึ้นคิดเป็นร้อยละ 383.34 และจากรูปภาพที่ 4.1.15 การจัดกลุ่มข้อมูลจะเห็นถึงการจัดกลุ่มข้อมูลที่ถูกต้องมากยิ่งขึ้น

4) ข้อมูลสังเคราะห์รูปร่างกลุ่มข้อมูลที่เรียงเป็นเส้นสามเส้น การตรวจจับข้อมูลผิดปกติด้วยอัลกอริทึม MOF ไม่ได้ส่งผลต่อการจัดกลุ่มด้วยวิธี k-mean มากนักโดยค่า Silhouette ของการจัดกลุ่มด้วยวิธี k-mean หลังการจำจัดข้อมูลผิดปกติมีค่าเพิ่มขึ้นเพียง 0.0138 เพิ่มขึ้นคิดเป็นร้อยละ 2.54 แต่ในทางกลับกันอัลกอริทึมการตรวจจับข้อมูลผิดปกติ MOF ส่งผลต่อการจัดกลุ่มด้วยวิธี agglomerative single linkage อย่างเห็นได้ชัด ค่า Silhouette ของการจัดกลุ่มด้วยวิธี agglomerative single linkage หลังการจำจัดข้อมูลผิดปกติมีค่าเพิ่มขึ้น 0.3399 เพิ่มขึ้นคิดเป็นร้อยละ 222.30 และจากรูปภาพที่ 4.1.17 การจัดกลุ่มข้อมูลจะเห็นถึงการจัดกลุ่มข้อมูลที่ถูกต้องมากยิ่งขึ้น

5) ข้อมูลสังเคราะห์รูปร่างกลุ่มข้อมูลที่กระจายเป็นกลุ่มใหญ่ การตรวจจับข้อมูลผิดปกติด้วยอัลกอริทึม MOF ไม่ได้ส่งผลต่อการจัดกลุ่มด้วยวิธี k-mean มากนักโดยค่า Silhouette ของการจัดกลุ่มด้วยวิธี k-mean หลังการจำจัดข้อมูลผิดปกติมีค่าเพิ่มขึ้นเพียง 0.009 เพิ่มขึ้นคิดเป็นร้อยละ 2.59 แต่ในทางกลับกันอัลกอริทึมการตรวจจับข้อมูลผิดปกติ MOF ส่งผลต่อการจัดกลุ่มด้วยวิธี agglomerative single linkage ค่า Silhouette ของการจัดกลุ่มด้วยวิธี agglomerative single linkage หลังการจำจัดข้อมูลผิดปกติมีค่าลดลง 0.1528 จากค่า Silhouette ที่ติดลบทำให้การจัดกลุ่มด้วยวิธี agglomerative single linkage ไม่เหมาะสมกับกลุ่มข้อมูลสังเคราะห์รูปร่างนี้

### 5.1.2 การจัดกลุ่มข้อมูลและตรวจจับข้อมูลผิดปกติกับข้อมูล spotify

1) การจัดกลุ่มข้อมูลด้วยวิธี agglomerative single linkage และตรวจจับข้อมูลผิดปกติด้วยวิธี MOF กับข้อมูล spotify นั้น ถึงแม้ว่าค่า Silhouette สูงสุดของการจัดกลุ่มแบบไม่กำจัดข้อมูลผิดปกติมีค่า 0.3398 มีค่ามากกว่าค่า Silhouette สูงสุดของการจัดกลุ่มแบบกำจัดข้อมูลผิดปกติมีค่า 0.2362 จึงกล่าวได้ว่าการกำจัดข้อมูลผิดปกติด้วยวิธีการ MOF แล้วจัดกลุ่มด้วยวิธี agglomerative single linkage นั้นไม่ได้ส่งผลต่อการจัดกลุ่มข้อมูล spotify นี้ให้ดีขึ้น เมื่อพิจารณาการวัดกราฟแสดงความสัมพันธ์ระหว่างลักษณะข้อมูลหลังการจัดกลุ่มแล้ว ดังภาพที่ 4.2.5 ก็พบว่าเป็นการจัดกลุ่มที่ยากต่อการตีความต่อ จึงสรุปได้ว่าการจัดกลุ่มแบบ agglomerative single linkage ทั้งแบบกำจัดข้อมูลผิดปกติและแบบไม่กำจัดข้อมูลผิดปกติ ไม่เหมาะสมสำหรับชุดข้อมูล spotify นี้

2) การจัดกลุ่มข้อมูลด้วยวิธี k-mean และตรวจจับข้อมูลผิดปกติกับข้อมูล spotify นั้นค่า Silhouette สูงสุดของการจัดกลุ่มแบบไม่กำจัดข้อมูลผิดปกติมีค่า 0.3534 มีค่ามากกว่าค่า Silhouette สูงสุดของการจัดกลุ่มแบบกำจัดข้อมูลผิดปกติมีค่า 0.3356 ซึ่งน้อยกว่าเพียงเล็กน้อย จึงกล่าวได้ว่าการกำจัดข้อมูลผิดปกติด้วยวิธีการ MOF แล้วจัดกลุ่มข้อมูลด้วยวิธีการ k-mean นั้นไม่ได้ส่งผลต่อการจัดกลุ่มข้อมูล spotify นี้ให้ดียิ่งขึ้น และเมื่อทำการวัดกราฟแสดงความสัมพันธ์ระหว่างลักษณะข้อมูลหลังการจัดกลุ่มแล้วดังภาพที่ 4.2.6 ก็พบว่ามีการจัดกลุ่มที่เหมาะสมแก่การตีความข้อมูลต่อ

### 5.1.3 การตีความการจัดกลุ่มข้อมูล spotify

โดยการจัดกลุ่ม spotify นี้ใช้วิธีการ k-mean หลังการกำจัดข้อมูลผิดปกติด้วยวิธี MOF โดยกำหนดจำนวนการจัดกลุ่มไว้ที่ 2 กลุ่ม แต่ละกลุ่มที่ถูกจัดมีจำนวนข้อมูลดังนี้ กลุ่มที่ 0 มีจำนวน 5,249 ตัว คิดเป็นร้อยละ 65.61 และกลุ่มที่ 1 มีจำนวน 2,751 ตัว คิดเป็นร้อยละ 34.39 เมื่อพิจารณาตามลักษณะที่เลือกมาในนี้จะสามารถตีความได้ดังต่อไปนี้

#### 1) ลักษณะ Danceability

กลุ่มที่ 0 ส่วนใหญ่การกระจายของข้อมูลจะอยู่ในช่วง 0.46 – 0.99 รวมเป็นร้อยละ 87.86 โดยอยู่ในช่วง 0.64 – 0.81 คิดเป็นร้อยละ 39.17 อยู่ในช่วง 0.46 – 0.64 คิดเป็นร้อยละ 29.68 และอยู่ในช่วง 0.81 – 0.99 คิดเป็นร้อยละ 19.01 กลุ่มที่ 1 ส่วนใหญ่การกระจายของข้อมูลจะอยู่ในช่วง 0.46 – 0.81 รวมเป็นร้อยละ 81.71 โดยอยู่ในช่วง 0.64 – 0.81 คิดเป็นร้อยละ 42.53 และอยู่ในช่วง 0.46 – 0.64 คิดเป็นร้อยละ 39.18

## 2) ลักษณะ energy

กลุ่มที่ 0 การกระจายของข้อมูลจะอยู่ในช่วง 0.44 – 1 รวมเป็นร้อยละ 92.07 โดยอยู่ในช่วง 0.81 – 1 คิดเป็นร้อยละ 35.38 อยู่ในช่วง 0.63 – 0.81 คิดเป็นร้อยละ 30.92 และอยู่ในช่วง 0.44 – 0.63 คิดเป็นร้อยละ 25.78 กลุ่มที่ 1 ส่วนใหญ่การกระจายของข้อมูลจะอยู่ในช่วง 0.63 – 1 รวมเป็นร้อยละ 88.37 โดยอยู่ในช่วง 0.81 – 1 คิดเป็นร้อยละ 67.90 และอยู่ในช่วง 0.63 – 0.81 คิดเป็นร้อยละ 20.46

## 3) ลักษณะ loudness

กลุ่มที่ 0 ส่วนใหญ่การกระจายของข้อมูลจะอยู่ในช่วง -11.61 ถึง -2.64 รวมเป็นร้อยละ 86.57 โดยอยู่ในช่วง -7.12 ถึง -2.64 คิดเป็นร้อยละ 56.52 และอยู่ในช่วง -11.61 ถึง -7.12 คิดเป็นร้อยละ 30.04 กลุ่มที่ 1 ส่วนใหญ่การกระจายของข้อมูลจะอยู่ในช่วง -11.61 ถึง -2.64 รวมเป็นร้อยละ 88.33 โดยอยู่ในช่วง -7.12 ถึง -2.64 คิดเป็นร้อยละ 47.87 และอยู่ในช่วง -11.61 ถึง -7.12 คิดเป็นร้อยละ 40.45

## 4) ลักษณะ speechiness

กลุ่มที่ 0 การกระจายของข้อมูลอยู่ในช่วง 0.02 – 0.37 รวมเป็นร้อยละ 90.53 โดยอยู่ในช่วง 0.02 – 0.2 คิดเป็นร้อยละ 63.48 และอยู่ในช่วง 0.2 – 0.37 คิดเป็นร้อยละ 27.05 กลุ่มที่ 1 ส่วนใหญ่การกระจายของข้อมูลจะอยู่ในช่วง 0.02 – 0.2 คิดเป็นร้อยละ 96.25

## 5) ลักษณะ acousticness

กลุ่มที่ 0 การกระจายของข้อมูลส่วนใหญ่อยู่ในช่วง 0 – 0.2 รวมเป็นร้อยละ 77.61 กลุ่มที่ 1 ส่วนใหญ่การกระจายของข้อมูลจะอยู่ในช่วง 0 – 0.2 คิดเป็นร้อยละ 95.31

## 6) ลักษณะ liveness

กลุ่มที่ 0 การกระจายของข้อมูลส่วนใหญ่อยู่ในช่วง 0.01 – 0.4 รวมเป็นร้อยละ 89.71 โดยอยู่ในช่วง 0.01 – 0.2 คิดเป็นร้อยละ 65.88 และอยู่ในช่วง 0.2 – 0.4 คิดเป็นร้อยละ 23.83 กลุ่มที่ 1 ส่วนใหญ่การกระจายของข้อมูลจะอยู่ในช่วง 0.01 – 0.4 คิดเป็นร้อยละ 87.24 โดยอยู่ในช่วง 0.01 – 0.2 คิดเป็นร้อยละ 68.59 และอยู่ในช่วง 0.2 – 0.4 คิดเป็นร้อยละ 18.65

## 7) ลักษณะ valence

กลุ่มที่ 0 การกระจายของข้อมูลส่วนใหญ่อยู่ในช่วง 0.02 – 0.59 รวมเป็นร้อยละ 75.75 โดยอยู่ในช่วง 0.21 – 0.4 คิดเป็นร้อยละ 28.06 อยู่ในช่วง 0.4 – 0.59 คิดเป็นร้อยละ 23.85 และอยู่ในช่วง 0.02 – 0.21 คิดเป็นร้อยละ 23.83 กลุ่มที่ 1 ส่วนใหญ่การกระจายของข้อมูลจะอยู่ในช่วง 0.02 – 0.4 คิดเป็นร้อยละ 75.43 โดยอยู่ในช่วง 0.02 – 0.21 คิดเป็นร้อยละ 50.96 และอยู่ในช่วง 0.21 – 0.4 คิดเป็นร้อยละ 24.46

#### 8) ลักษณะ tempo

กลุ่มที่ 0 การกระจายของข้อมูลส่วนใหญ่อยู่ในช่วง 95.98 – 189.07 รวมเป็นร้อยละ 88.07 โดยอยู่ในช่วง 127.01 – 158.04 คิดเป็นร้อยละ 43.63 อยู่ในช่วง 158.04 – 189.07 คิดเป็นร้อยละ 26.25 และอยู่ในช่วง 95.98 – 127.01 คิดเป็นร้อยละ 18.19 กลุ่มที่ 1 ส่วนใหญ่การกระจายของข้อมูลจะอยู่ในช่วง 95.98 – 158.04 คิดเป็นร้อยละ 84.30 โดยอยู่ในช่วง 127.01 – 158.04 คิดเป็นร้อยละ 60.55 และอยู่ในช่วง 95.98 – 127.01 คิดเป็นร้อยละ 23.73

#### 9) ลักษณะ instrumentalness

กลุ่มที่ 0 การกระจายของข้อมูลส่วนใหญ่อยู่ในช่วง 0 – 0.2 คิดเป็นร้อยละ 93.18 กลุ่มที่ 1 ส่วนใหญ่การกระจายของข้อมูลจะอยู่ในช่วง 0.59 – 0.99 คิดเป็นร้อยละ 86.99 โดยอยู่ในช่วง 0.79 – 0.99 คิดเป็นร้อยละ 62.85 และอยู่ในช่วง 0.59 – 0.79 คิดเป็นร้อยละ 24.14

#### 10) ประเภทของเพลง

กลุ่มที่ 0 การกระจายของข้อมูลส่วนใหญ่จะอยู่ในประเภทเพลง Underground Rap, Dark Trap, Hiphop, trap, RnB, Trap Metal และ Rap รวมเป็นร้อยละ 75.86 โดยแบ่งเป็น Underground Rap คิดเป็นร้อยละ 21.79 Dark Trap คิดเป็นร้อยละ 12.02 Hiphop คิดเป็นร้อยละ 11.32 trap คิดเป็นร้อยละ 8.74 RnB คิดเป็นร้อยละ 7.96 Trap Metal คิดเป็นร้อยละ 7.03 และ Rap คิดเป็นร้อยละ 6.99 กลุ่มที่ 1 การกระจายของข้อมูลส่วนใหญ่จะอยู่ในประเภทเพลง techno, psytrance, techhouse, trance และ dnb รวมเป็นร้อยละ 81.10 โดยแบ่งเป็น techno คิดเป็นร้อยละ 21.08 psytrance คิดเป็นร้อยละ 20.76 techhouse คิดเป็นร้อยละ 14.14 trance คิดเป็นร้อยละ 12.65 และ dnb คิดเป็นร้อยละ 12.47

## 5.2 ข้อเสนอแนะ

5.2.1 นำการตรวจจับข้อมูลผิดปกติ MOF ทดสอบร่วมกับอัลกอริทึมการจัดกลุ่มอื่นๆ โดยทดสอบกับข้อมูลสังเคราะห์ที่หลากหลายรูปร่างและทดสอบกับข้อมูลโลจิริง เพื่อตรวจสอบการทำงานร่วมกันระหว่างอัลกอริทึมการจัดกลุ่มต่างๆกับอัลกอริทึม MOF

5.2.2 ทดสอบวิธีการตรวจจับข้อมูลผิดปกติหลายแบบร่วมกับอัลกอริทึมการจัดกลุ่มเพื่อที่จะให้เห็นถึงการทำงานที่หลากหลาย และดูความเหมาะสมของอัลกอริทึมที่ใช้กับข้อมูลรูปแบบต่างๆ

## รายการอ้างอิง

- [1] Medium. การทำ Machine Learning ด้วย Clustering Model. [Online]. 2020, Available form : <https://medium.com/tnt-university/การทำ-machine-learning-ด้วย-clustering-model2a3c392e7faa>.
- [2] Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys*, 31(3), 264-323
- [3] Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.
- [4] Gan, G., Ma, C., & Wu, J. (2007). *Data Clustering: Theory, Algorithms, and Applications* (Vol. 20). SIAM
- [5] Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20, 53-65.
- [6] scikit-learn [Online]. 2023, Available form: [https://scikit-learn.org/stable/auto\\_examples/cluster/plot\\_cluster\\_comparison.html](https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html)
- [7] Spotify. [Online]. 2018, Available form : <https://developer.spotify.com/documentation/web-api>
- [8] Sampling Techniques. [Online]. 2015, Available form: <https://www.healthknowledge.org.uk/public-health-textbook/research-methods/1a-epidemiology/methods-of-sampling-population>
- [9] What Is Percentile in Statistics and How to Calculate it?. [Online]. 2023, Available form: <https://www.simplilearn.com/tutorials/data-analytics-tutorial/percentile>
- [10] Phichapop Changsakul, Somjai Boonsiri and Krung Sinapiromsaran, "Mass-ratio-variance based Outlier Factor," 2021 18th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2021, p. 1-5, DOI : 10.1109/JCSSE53117.2021.9493811.

## ภาคผนวก

## ภาคผนวก ก

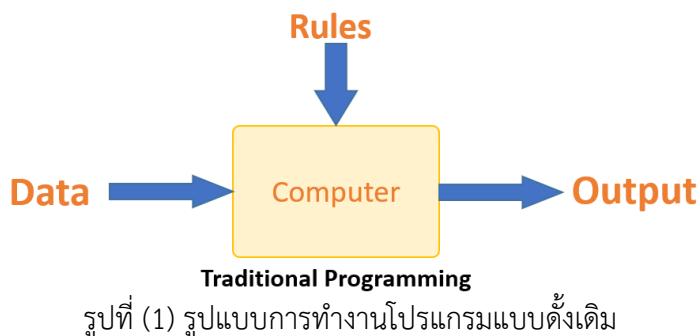
### แบบเสนอหัวข้อโครงการ รายวิชา 2301399 Project Proposal

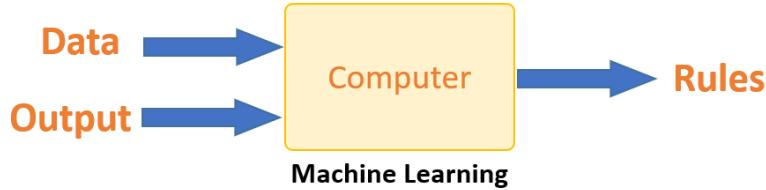
#### ปีการศึกษา 2564

ชื่อโครงการ (ภาษาไทย)	ขั้นตอนวิธีการเก็บกลุ่มอัลเอน์โอลเอฟ
ชื่อโครงการ (ภาษาอังกฤษ)	MOF Clustering algorithm
อาจารย์ที่ปรึกษา	รองศาสตราจารย์ ดร. กรุง สินอภิรมย์สรายุ
ผู้ดำเนินการ	นายจิรภัทร ชัยบุตร เลขประจำตัวนิสิต 6234307323 สาขาวิชา คณิตศาสตร์ ภาควิชาคณิตศาสตร์และวิทยาคอมพิวเตอร์ คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

#### หลักการและเหตุผล

ในทางวิทยาการคอมพิวเตอร์ ศาสตร์ทางด้านการเรียนรู้ของเครื่องจักร (Machine Learning) จัดเป็นส่วนหนึ่งของปัญญาประดิษฐ์ที่นักวิทยาการคอมพิวเตอร์ต้องการสร้างขั้นตอนวิธีการเรียนรู้จากข้อมูลในอดีตด้วยตัวแบบ (Model) ที่อาจอยู่ในรูปของสมการ โครงสร้างต้นไม้ การเขียนโดยเครือข่ายแล้วนำไปประยุกต์กับข้อมูลที่ยังไม่พบในอนาคตโดยความแตกต่างระหว่างการพัฒนาโปรแกรมเพื่อแก้ปัญหาการคำนวณ กับการพัฒนาโปรแกรมที่เรียนรู้ได้ด้วยเครื่องจักรคือ การเขียนโปรแกรมเพื่อแก้ปัญหาการคำนวณ มีการกำหนดกฎกติกาและขั้นตอนการประมวลผลเพื่อให้ได้ผลลัพธ์ที่ต้องการ เมื่อมีข้อมูลที่ป้อนเข้ามาตรงกับกฎที่สร้างไว้โปรแกรมจะทำการประมวลผลแล้วให้ผลลัพธ์ตามที่กำหนดไว้แต่แรกดูรูปที่ (1) ถ้ามีการเพิ่มระบบที่ซับซ้อนขึ้นก็มีการพัฒนาเพิ่มตามแต่สำหรับการเรียนรู้ด้วยเครื่องจักรจะไม่มีการกำหนดกฎกติกาและรูปแบบการคำนวณที่ชัดเจนไว้ แต่จะมีการใช้ตัวแบบที่มีพารามิเตอร์ซึ่งจะถูกเปลี่ยนตามข้อมูลในอดีตทำให้เกิดการสร้างกฎใหม่ขึ้นมาเองได้ตามรูปที่ (2) ผู้พัฒนาโปรแกรมไม่จำเป็นต้องเขียนกฎใหม่ทุกครั้งที่มีข้อมูลใหม่เพิ่มมา ขั้นตอนวิธีการเรียนรู้จะปรับพารามิเตอร์ให้เข้ากับข้อมูลใหม่่องอัตโนมัติ [1]





รูปที่ (2) รูปแบบการทำงานการเรียนรู้ด้วยเครื่องจักร  
การเรียนรู้ด้วยเครื่องจักร (Machine learning) แบ่งออกเป็นได้ 3 ประเภท ได้แก่

1. การเรียนรู้แบบมีผู้สอน (Supervised learning) จะใช้ข้อมูลในการปรับพารามิเตอร์เรียก ข้อมูลสอน (Training data) ตัวแบบจะทำนายค่าของตัวอย่าง แล้วนำมาปรับปรุงพารามิเตอร์ให้เหมาะสม ในกรณีที่การทำนายมีความผิดพลาด การเรียนรู้แบบมีผู้สอนแยกออกเป็น 2 ประเภทย่อย

- การจำแนกประเภท (Classification) ตัวแปรเป้าหมายของแต่ละตัวอย่างมีค่าไม่ต่อเนื่อง โดยตัวจำแนกประเภท (Classifier) จะทำนายค่าความน่าจะเป็นที่ตัวอย่างดังกล่าวจะเป็นชนิดที่กำหนดให้
- สมการการถดถอย (Regression) โดยที่ตัวแปรเป้าหมายมีค่าต่อเนื่อง ยกตัวอย่างเช่น การทำนายมูลค่าของหุ้น

ตัวอย่างตัวแบบของการเรียนรู้แบบมีผู้สอน เช่น สมการถดถอยเชิงเส้น (Linear Regression), สมการถดถอยโลจิสติก (Logistic Regression), ต้นไม้การตัดสินใจ (Decision Tree), ป่าสุ่ม (Random Forest) และเครือข่ายประสาท (Neural network) เป็นต้น

2. การเรียนรู้แบบไม่มีผู้สอน (Unsupervised learning) จะไม่มีการทำหนดตัวแปรเป้าหมาย การเรียนรู้ที่เกิดขึ้นการจัดกลุ่มของข้อมูลแยกเป็นแต่ละส่วน เช่น การแยกข้อมูลประชากรออกเป็นส่วน ๆ ที่สัมพันธ์กัน การใช้งานหลักมี 2 อย่าง [4] คือ

- การลดมิติของข้อมูล (Dimensionality reduction) เน้นการลดความซับซ้อนของปริมาณข้อมูล ก่อนนำไปใช้ เพื่อนำมาแสดงในกราฟที่มีมิติที่ต่ำลงได้
- การวิเคราะห์การเกาะกลุ่ม (Clustering analysis) เน้นการรวมข้อมูลเป็นกลุ่ม ตามลักษณะต่าง ๆ ของตัวอย่าง เช่น การจัดกลุ่มลูกค้าตามพฤติกรรมการซื้อ

ตัวอย่างของการเรียนรู้แบบไม่มีผู้สอน เช่น การเกาะกลุ่มค่าเฉลี่ยเค (K-means clustering), ตัวแบบผสมเกาส์ (Gaussian mixture model), การเกาะกลุ่มตามลำดับชั้น (Hierarchical cluster) และ การวิเคราะห์แคนนอนสำคัญ (Principal component analysis) เป็นต้น

3. การเรียนรู้แบบเสริมกำลัง (Reinforcement learning) เน้นการเรียนรู้วิธีการตัดสินใจภายใต้สถานการณ์ต่าง ๆ โดยมีเป้าหมายสุดท้ายคือรางวัลที่เกิดการเลือกการกระทำการทำงาน ทำให้เกิดการค้นหาแนวทางรับมือกับปัญหาที่ดีที่สุด ตัวอย่างการใช้งาน เช่น การเล่นโกะของ AlphaGO, รถที่ขับเคลื่อนได้ด้วยตนเอง (Self-driving car) และบอทแลกเปลี่ยนสต็อก (Stock trading bot) เป็นต้น

สำหรับโครงการนี้ เลือกการพัฒนาขั้นตอนวิธีการเกาะกลุ่มรูปแบบใหม่ของการเรียนรู้ด้วยเครื่องจักร ซึ่งจัดอยู่ในประเภทของการเรียนรู้แบบไม่มีผู้สอน โดยใช้หลักเกณฑ์ในการรวมกลุ่มเชิงความหนาแน่นตามค่า MOF จุดที่มีค่า MOF score สูงจะไม่ถูกนำมารวมกับจุดอื่น ๆ แต่จุดที่มีค่า MOF score ต่ำจะถูกนำมาเข้ามาร่วมกันเป็นกลุ่ม โดยขอบเขตการเกาะกลุ่มนั้นจะพิจารณาจากค่าเฉลี่ยของสัดส่วนความหนาแน่นแต่ละจุดนำมาเข้มต่อกันเป็นกลุ่ม

### วัตถุประสงค์

พัฒนาโปรแกรมการเรียนรู้ด้วยเครื่องจักรด้วยภาษา Python สำหรับขั้นตอนวิธีการเกาะกลุ่มเอ็มโอเอฟ ที่ไม่ใช้พารามิเตอร์ และทดสอบโปรแกรมกับข้อมูลการเกาะกลุ่มสังเคราะห์ห้ารูปแบบ และข้อมูลจริงจาก Kaggle

### ขอบเขตของโครงการ

ในโครงการนี้พัฒนาโปรแกรมภาษา Python สำหรับขั้นตอนวิธีการเกาะกลุ่มเอ็มโอเอฟ พร้อมทดสอบประสิทธิภาพของโปรแกรมกับข้อมูลสังเคราะห์ห้ารูปแบบ (1) วงกลมซ้อนสองวง (2) รูปพระจันทร์ซ้อนสองวง (3) กลุ่มข้อมูลกระจายสามกลุ่ม (4) กลุ่มข้อมูลที่เรียงเป็นเส้นสี่เส้น (5) กลุ่มข้อมูลที่กระจายเป็นกลุ่มใหญ่ และทดสอบกับข้อมูลโลกจริงจาก Kaggle ขนาดข้อมูล 42,306 ตัว

### วิธีการดำเนินงาน

1. ศึกษาปัญหา และกำหนดหัวข้อที่จะทำโครงการ
2. ศึกษาขั้นตอนวิธี Mass-ratio-variance Outlier Factor (MOF) และ Clustering algorithm และงานวิจัยที่เกี่ยวข้อง
3. ออกแบบขั้นตอนวิธีการเกาะกลุ่มเอ็มโอเอฟ
4. พัฒนาโปรแกรมโดยใช้ภาษา Python ทดลองกับข้อมูลที่สังเคราะห์ขึ้นมาและข้อมูลโลกจริง
5. วิเคราะห์ผลลัพธ์พร้อมสรุป
6. จัดทำรูปเล่มโครงการ
7. นำเสนอผลการดำเนินงานวิธีดำเนินงาน

วิธีดำเนินงาน	สิงหาคม 2565 - เมษายน 2566								
	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.
1. ศึกษาปัญหา และกำหนดหัวข้อที่จะทำโครงการ									
2. ศึกษาขั้นตอนวิธี Mass-ratio-variance Outlier Factor (MOF) และ Clustering algorithm และงานวิจัยที่เกี่ยวข้อง									
3. ออกแบบขั้นตอนวิธีการเกาะกลุ่มเอ็มโอดอฟ									
4. พัฒนาโปรแกรมโดยใช้ภาษา Python ทดลอง กับข้อมูลที่สังเคราะห์ขึ้นมาและข้อมูลโลกจริง									
5. วิเคราะห์ผลลัพธ์พร้อมสรุป									
6. จัดทำรูปเล่มโครงการ									
7. นำเสนอผลการดำเนินงานวิธีดำเนินงาน									

### ประโยชน์ที่คาดว่าจะได้รับ

สามารถจัดการกลุ่มข้อมูลด้วยวิธี MOF Clustering ทั้งจากตัวข้อมูลที่สังเคราะห์ขึ้นมาและข้อมูลโลกจริง

### อุปกรณ์และเครื่องมือที่ใช้

- Colaboratory บน Google cloud
- โน๊ตบุ๊ค Lenovo legion 5 AMD 16 GB DDR4 512GB SSD PCIe

## ภาคผนวก ข

### โปรแกรมและตัวอย่างโค้ด

โปรแกรมที่ใช้ในการเขียนภาษา Python มีหลากหลายตัวเลือก ในโครงการนี้ได้เลือกใช้ Google colab โดยโค้ดมีดังต่อไปนี้

#### 1. นิยามการตรวจจับข้อมูลผิดปกติจากอัลกอริทึม MOF

```

def euclid_distance(p1, p2):
    return np.sqrt(np.sum(np.square(p1 - p2)))
def Neighborhood(arr):
    arr_size = len(arr)
    n_arr = arr.reshape((1, arr_size))
    distance_diff = n_arr - n_arr.T
    nbh = np.sum(np.where(distance_diff >= 0, 1, 0), axis=0)
    return nbh
def NBH_Matrix(data):
    d_size = len(data)
    dist_matrix = np.zeros((d_size, d_size))
    for i in range(d_size):
        for j in range(i+1, d_size):
            dist_matrix[i,j] = euclid_distance(data[i], data[j])
            dist_matrix[j,i] = dist_matrix[i,j]
    Neighbor_matrix = np.ones((d_size, d_size))
    Neighbor_matrix = np.apply_along_axis(Neighborhood, 1, dist_matrix)
    return Neighbor_matrix
def MassRatio(data):
    minor_NBH_matrix = NBH_Matrix(data)
    return minor_NBH_matrix / np.transpose(minor_NBH_matrix)
def MOF_p(pre_arr, i):
    arr = np.delete(pre_arr, i, axis=0)
    return np.var(arr)
def MOF(data):
    MR_Matrix = MassRatio(data)
    d_size = len(data)
    MRV_Matrix = np.zeros(d_size)
    for i in range(d_size):
        MRV_Matrix[i] = MOF_p(MR_Matrix[:, i], i)
    return MRV_Matrix

```

ภาพที่ ข.1 การคำนวณค่า MOF

```

def find_outliers(data, percentile=95):
    # Calculate MOF values
    mof_values = MOF(data)
    # Determine the threshold based on the specified percentile
    threshold = np.percentile(mof_values, percentile)
    # Find outliers based on the calculated threshold
    outliers = np.where(mof_values > threshold)
    return outliers

def remove_outliers(data, percentile=95):
    # Find outlier indices using the find_outliers function
    outlier_indices = find_outliers(data, percentile)
    # Remove outliers from the data
    cleaned_data = np.delete(data, outlier_indices, axis=0)
    # Get the indices of the cleaned data in the original dataset
    cleaned_indices = np.setdiff1d(np.arange(data.shape[0]), outlier_indices)
    return cleaned_data, cleaned_indices

```

ภาพที่ ข.2 เงื่อนไขการตรวจจับข้อมูลผิดปกติจากค่า MOF

จากการที่ ข.1 และข.2 ทำการคำนวณค่า MOF และหาตัวข้อมูลผิดปกติจากการตั้งเกณฑ์การตรวจจับที่ เปอร์เซ็นไทล์ 95

## 2. นิยามการทำงานอัลกอริทึม K-mean สำหรับข้อมูลสังเคราะห์

```

def k_means_clustering(data, num_clusters):
    kmeans = KMeans(n_clusters=num_clusters, n_init=10, random_state=42)

    # Fit the model to the data
    kmeans.fit(data)

    # Get the cluster assignments for each data point
    cluster_assignments = kmeans.labels_

    # Get the cluster centroids
    cluster_centroids = kmeans.cluster_centers_

    return cluster_assignments, cluster_centroids

```

ภาพที่ ข.3 การทำงานอัลกอริทึม K-mean สำหรับข้อมูลสังเคราะห์

### 3. นิยามการทำงานอัลกอริทึม Agglomerative single linkage สำหรับข้อมูลสังเคราะห์

```

def plot_dendrogram(linkage_matrix, **kwargs):
    # Plot the corresponding dendrogram
    dendrogram(linkage_matrix, **kwargs)
def hierarchical_clustering(data, n_clusters, linkage_method='single', plot_dendro=False):
    # Compute the linkage matrix
    linkage_matrix = linkage(data, method=linkage_method)
    model = AgglomerativeClustering(n_clusters=n_clusters, linkage=linkage_method, distance_threshold=None)
    # Fit the model to the data
    model = model.fit(data)
    if plot_dendro:
        plt.figure(figsize=(12, 6))
        plot_dendrogram(linkage_matrix, truncate_mode='level', p=400)
        plt.xlabel("Index of data points")
        plt.ylabel("Euclidean distance")
        plt.show()
    # Return the cluster labels for each data point
    return model.labels_
# Visualization function
def plot_clusters(data, cleaned_data, cleaned_indices, labels):
    plt.scatter(cleaned_data[:, 0], cleaned_data[:, 1], c=labels, cmap='viridis', s=10)

    outlier_indices = np.setdiff1d(np.arange(data.shape[0]), cleaned_indices)
    plt.scatter(data[outlier_indices, 0], data[outlier_indices, 1], c='red', marker='x', s=50, label='Outliers')

    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.legend()
    plt.show()

```

ภาพที่ ๗.๔ การทำงานอัลกอริทึม Agglomerative single linkage การวาดภาพ dendrogram และการวาด  
กราฟแสดงผล

### 4. การสังเคราะห์ข้อมูลรูปร่างต่างๆและการตรวจจับข้อมูลผิดปกติแสดงผลในรูปของกราฟ และ ค่า silhouette สำหรับอัลกอริทึม K-mean

```

data1, labels1 = make_blobs(n_samples=400, centers=3, random_state=42, cluster_std=2.0)
data2, labels2 = make_circles(n_samples=400, factor=0.5, noise=0.05, random_state=42)
data3, labels3 = make_blobs(n_samples=400, centers=1, cluster_std=5.0, random_state=42)
data4, labels4 = make_moons(n_samples=400, noise=0.1, random_state=42)
data5, labels5 = make_blobs(n_samples=500, centers=3, random_state=42, cluster_std=2.0)
transformation = np.array([[0.6, -0.6], [-0.4, 0.8]])
data = np.dot(data5, transformation)

```

ภาพที่ ๗.๕ การสังเคราะห์รูปร่างต่างๆสำหรับอัลกอริทึม K-mean

```

cleaned_data, cleaned_indices = remove_outliers(data, percentile=95)
outlier_indices = np.setdiff1d(np.arange(data.shape[0]), cleaned_indices)
optimal_clusters = 2
# Apply k-means clustering to the original data
cluster_assignments_original, cluster_centroids_original = k_means_clustering(data, optimal_clusters)
# Apply k-means clustering to the cleaned data
cluster_assignments_cleaned, cluster_centroids_cleaned = k_means_clustering(cleaned_data, optimal_clusters)
# Plot the original data
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.scatter(data[:, 0], data[:, 1], c=cluster_assignments_original, cmap='viridis')
plt.scatter(cluster_centroids_original[:, 0], cluster_centroids_original[:, 1], c='red', marker='x', s=100, label='Centroids')
plt.title('K-means Clustering (Original Data)')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
# Plot the cleaned data
plt.subplot(1, 2, 2)
plt.scatter(cleaned_data[:, 0], cleaned_data[:, 1], c=cluster_assignments_cleaned, cmap='viridis')
plt.scatter(cluster_centroids_cleaned[:, 0], cluster_centroids_cleaned[:, 1], c='red', marker='x', s=100, label='Centroids')
plt.scatter(data[outlier_indices, 0], data[outlier_indices, 1], c='black', marker='x', s=100, label='Outliers')
plt.title('K-means Clustering (Cleaned Data)')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.show()
# Calculate the silhouette score for the original data
silhouette_score_original = silhouette_score(data, cluster_assignments_original)
# Calculate the silhouette score for the cleaned data
silhouette_score_cleaned = silhouette_score(cleaned_data, cluster_assignments_cleaned)
print("Silhouette score (original data):", silhouette_score_original)
print("Silhouette score (cleaned data):", silhouette_score_cleaned)

```

ภาพที่ ข.6 การตรวจจับข้อมูลผิดปกติแสดงผลในรูปของกราฟและค่า silhouette สำหรับอัลกอริทึม K-mean

## 5. การสังเคราะห์ข้อมูลรูปร่างต่างๆและการตรวจจับข้อมูลผิดปกติแสดงผลในรูปของกราฟ และค่า silhouette สำหรับอัลกอริทึม Agglomerative single linkage

```

# Generate datasets
n_samples = 400
noisy_circles = datasets.make_circles(n_samples=n_samples, factor=0.5, noise=0.05)
noisy_moons = datasets.make_moons(n_samples=n_samples, noise=0.05)
blobs = datasets.make_blobs(n_samples=n_samples, random_state=8)
no_structure = np.random.rand(n_samples, 2), None

random_state = 170
X, y = datasets.make_blobs(n_samples=n_samples, random_state=random_state)
transformation = [[0.6, -0.6], [-0.4, 0.8]]
X_aniso = np.dot(X, transformation)
aniso = (X_aniso, y)

varied = datasets.make_blobs(
    n_samples=n_samples, cluster_std=[1.0, 2.5, 0.5], random_state=random_state
)

datasets = [noisy_circles, noisy_moons, blobs, no_structure, aniso, varied]

```

ภาพที่ ข.7 การสังเคราะห์รูปร่างต่างๆสำหรับอัลกอริทึม Agglomerative single linkage

```

# Loop through each dataset, perform clustering, and visualize the results
for i, dataset in enumerate(datasets):
    print(f"Dataset {i + 1}:")
    X, y = dataset

    # Set a flag to indicate whether the dataset is either noisy_circles or noisy_moons
    is_noisy_circles_or_moons = i in [0, 1]

    n_clusters = 2 if is_noisy_circles_or_moons else 3

    print("Without eliminating outliers:")
    labels_without_outliers_removed = hierarchical_clustering(X, n_clusters=n_clusters, linkage_method='single', plot_dendro=True)
    plot_clusters(X, X, np.arange(X.shape[0]), labels_without_outliers_removed)
    silhouette_score_without_outliers_removed = silhouette_score(X, labels_without_outliers_removed)
    print(f"Silhouette Score: {silhouette_score_without_outliers_removed:.4f}")

    print("With eliminating outliers:")
    cleaned_data, cleaned_indices = remove_outliers(X, percentile=95)
    labels_with_outliers_removed = hierarchical_clustering(cleaned_data, n_clusters=n_clusters, linkage_method='single', plot_dendro=True)

    plot_clusters(X, cleaned_data, cleaned_indices, labels_with_outliers_removed)
    silhouette_score_with_outliers_removed = silhouette_score(cleaned_data, labels_with_outliers_removed)
    print(f"Silhouette Score: {silhouette_score_with_outliers_removed:.4f}")

```

ภาพที่ ข.8 การตรวจจับข้อมูลผิดปกติแสดงผลในรูปของกราฟและค่า silhouette สำหรับยัลกอยริทึม

Agglomerative single linkage

## 6. นิยามการทำงานต่างๆเพื่อใช้กับข้อมูล spotify

```

def systematic_sampling(data, n_samples):
    """
        data: pandas DataFrame containing the dataset
        n_samples: number of samples to be drawn from the dataset
    """
    data_size = len(data)
    k = data_size // n_samples
    random_start = np.random.randint(0, k)
    indices = np.arange(random_start, data_size, k)[:n_samples]
    sampled_data = data.iloc[indices]

    # Reset the index and drop the old index column
    sampled_data = sampled_data.reset_index(drop=True)

    return sampled_data

```

ภาพที่ ข.9 การสุ่มข้อมูลแบบ systematic

```

def calculate_silhouette_scores(data):
    silhouette_scores = []
    cluster_range = range(2, 11)

    for k in cluster_range:
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(data)
        sil_score = silhouette_score(data, kmeans.labels_)
        silhouette_scores.append(sil_score)

    return silhouette_scores

```

ภาพที่ ข.10 คำนวณค่า silhouette สำหรับอัลกอริทึม K-mean

```

def plot_attribute_bars(data, attribute, cluster_col='cluster', bins=5):
    # Create a temporary DataFrame with the attribute and cluster columns
    temp_df = data[[attribute, cluster_col]]

    # Calculate the minimum and maximum values of the attribute
    min_val = temp_df[attribute].min()
    max_val = temp_df[attribute].max()

    # Calculate the bin width
    bin_width = (max_val - min_val) / bins

    # Create bins
    bin_edges = np.arange(min_val, max_val + bin_width, bin_width)
    bin_labels = [f'{round(edge, 2)} - {round(edge + bin_width, 2)}' for edge in bin_edges[:-1]]

    # Assign each data point to a bin
    temp_df['bin'] = pd.cut(temp_df[attribute], bins=bin_edges, labels=bin_labels, include_lowest=True)

    # Count the number of data points in each bin for each cluster
    bin_counts = temp_df.groupby([cluster_col, 'bin']).size().unstack().T

    # Plot the bar graph
    ax = bin_counts.plot(kind='bar', figsize=(10, 5))
    ax.set_xlabel(f'{attribute} ranges')
    ax.set_ylabel('Count')
    ax.set_title(f'{attribute} Distribution by Cluster')
    plt.xticks(rotation=45)
    plt.legend(title='Cluster', loc='upper right')
    plt.show()

```

ภาพที่ ข.11 แสดงผลการจัดกลุ่มตามแต่ละลักษณะที่เลือกในรูปของกราฟแท่ง

```

def print_attribute_bin_counts(data, attribute, cluster_col='cluster', bins=5):
    # Create a temporary DataFrame with the attribute and cluster columns
    temp_df = data[[attribute, cluster_col]]

    # Calculate the minimum and maximum values of the attribute
    min_val = temp_df[attribute].min()
    max_val = temp_df[attribute].max()

    # Calculate the bin width
    bin_width = (max_val - min_val) / bins

    # Create bins
    bin_edges = np.arange(min_val, max_val + bin_width, bin_width)
    bin_labels = [f'{round(edge, 2)} - {round(edge + bin_width, 2)}' for edge in bin_edges[:-1]]

    # Assign each data point to a bin
    temp_df['bin'] = pd.cut(temp_df[attribute], bins=bin_edges, labels=bin_labels, include_lowest=True)

    # Count the number of data points in each bin for each cluster
    bin_counts = temp_df.groupby([cluster_col, 'bin']).size().unstack().T

    # Print the bin counts for each attribute, sorted by cluster
    print(f'{attribute} Distribution by Cluster:')
    print(bin_counts)
    print()

```

ภาพที่ ข.12 แสดงผลการจัดกลุ่มตามแต่ละลักษณะที่เลือกเป็นตัวเลข

```

def plot_genre_bars(data, genre_col='genre', cluster_col='cluster'):
    # Create a temporary DataFrame with the genre and cluster columns
    temp_df = data[[genre_col, cluster_col]]

    # Count the number of data points in each genre for each cluster
    genre_counts = temp_df.groupby([cluster_col, genre_col]).size().unstack().T

    # Plot the bar graph
    ax = genre_counts.plot(kind='bar', figsize=(15, 7))
    ax.set_xlabel('Genres')
    ax.set_ylabel('Count')
    ax.set_title('Genre Distribution by Cluster')
    plt.xticks(rotation=45)
    plt.legend(title='Cluster', loc='upper right')
    plt.show()

```

ภาพที่ ข.13 แสดงผลการจัดกลุ่มตามประเภทของเพลงในรูปของกราฟแท่ง

```

def calculate_wcss(data):
    wcss = []
    cluster_range = range(2, 11)

    for k in cluster_range:
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(data)
        wcss.append(kmeans.inertia_)

    return wcss

```

ภาพที่ ข.14 คำนวณค่า WCSS เพื่อใช้แสดงผลกราฟ elbow

## 7. การจัดกลุ่มข้อมูล spotify ด้วยอัลกอริทึม k-mean และอัลกอริทึม Agglomerative single linkage

```

uploaded = files.upload()
dataset_path = "spotify_dataset10.csv"
data = pd.read_csv(dataset_path)
n_samples = 8000
sampled_data = systematic_sampling(data, n_samples)
print(sampled_data)

```

ภาพที่ ข.15 อัปโหลดไฟล์และสุ่มข้อมูลโดยสุ่มแบบ systematic

จากภาพที่ ข.15 อัปโหลดไฟล์ spotify และสุ่มข้อมูลจำนวน 8,000 ตัวจากนิยามการสุ่มข้อมูลแบบ systematic ภาพที่ ข.9

```

selected_columns = ['danceability', 'energy', 'loudness', 'speechiness', 'acousticness', 'liveness', 'valence', 'tempo', 'instrumentalness']
data_selected = data_clean[selected_columns].values

cleaned_data, cleaned_indices = remove_outliers(data_selected, percentile=95)

# Standardize the data
scaler = StandardScaler()
cleaned_scaled_data = scaler.fit_transform(cleaned_data)

```

ภาพที่ ข.16 เลือกลักษณะที่ทำการจัดกลุ่ม กำจัดข้อมูลผิดปกติ และสเกลข้อมูล

จากภาพที่ ข.16 ทำการเลือกลักษณะที่ต้องการจะจัดกลุ่ม แล้วทำการกำจัดข้อมูลผิดปกติ และสเกลข้อมูลตามลำดับ

```

linked = linkage(cleaned_scaled_data, method='single')
plt.figure(figsize=(12, 6))
dendrogram(linked, truncate_mode='level', p=5)
plt.title('Hierarchical Clustering Dendrogram (Single)')
plt.xlabel('Index')
plt.ylabel('Single Linkage Distance')
plt.show()

```

ภาพที่ ข.17 แสดงแผนภาพ Dendrogram จากวิธีการจัดกลุ่ม Agglomerative single linkage

```

wcss_full = calculate_wcss(data_scaled)
wcss_cleaned = calculate_wcss(cleaned_scaled_data)

plt.figure(figsize=(10, 5))
plt.plot(cluster_range, wcss_full, marker='o', linestyle='--', label="Without Outlier Elimination")
plt.plot(cluster_range, wcss_cleaned, marker='o', linestyle='--', label="With Outlier Elimination")
plt.xlabel("Number of clusters")
plt.ylabel("WCSS (Within-Cluster Sum of Squares)")
plt.title("Elbow Plot: with vs. Without Outlier Elimination")
plt.grid()
plt.legend()
plt.show()

```

ภาพที่ ข.18 แสดงกราฟ elbow สำหรับอัลกอริทึม k-mean

```

scaler = StandardScaler()
data_scaled = scaler.fit_transform(data_selected)
silhouette_scores_original = []
silhouette_scores_cleaned = []
for n_clusters in range(2, 11):
    # Apply AgglomerativeClustering for both cases
    model_original = AgglomerativeClustering(n_clusters=n_clusters, linkage='single')
    model_original.fit(data_scaled)
    cluster_labels_original = model_original.labels_

    model_cleaned = AgglomerativeClustering(n_clusters=n_clusters, linkage='single')
    model_cleaned.fit(cleaned_scaled_data)
    cluster_labels_cleaned = model_cleaned.labels_
    # Calculate silhouette scores for both cases
    silhouette_avg_original = silhouette_score(data_scaled, cluster_labels_original)
    silhouette_avg_cleaned = silhouette_score(cleaned_scaled_data, cluster_labels_cleaned)

    # Append silhouette scores to the respective lists
    silhouette_scores_original.append(silhouette_avg_original)
    silhouette_scores_cleaned.append(silhouette_avg_cleaned)

```

ภาพที่ ข.19 คำนวณค่า silhouette จากการทำจัดข้อมูลผิดปกติและไม่ทำจัดข้อมูลผิดปกติ

```
# Create a pandas DataFrame to display the results
results_df = pd.DataFrame([
    'Number of Clusters': range(2, 11),
    'Silhouette Score (Original Data)': silhouette_scores_original,
    'Silhouette Score (Cleaned Data)': silhouette_scores_cleaned
])
# Display the results in table form
print(results_df)
cluster_range = range(2, 11)
plt.figure(figsize=(10, 5))
plt.plot(cluster_range, silhouette_scores_original, marker='o', linestyle='--', label="Without Outlier Elimination")
plt.plot(cluster_range, silhouette_scores_cleaned, marker='o', linestyle='--', label="With Outlier Elimination")
plt.xlabel("Number of clusters")
plt.ylabel("Silhouette Score")
plt.title("Silhouette Scores: With vs. Without Outlier Elimination (Agglomerative clustering)")
plt.grid()
plt.legend()
plt.show()
```

ภาพที่ ข.20 แสดงผลค่า silhouette จากการจำจัดข้อมูลผิดปกติและไม่จำจัดข้อมูลผิดปกติในรูปของตัวเลขและรูปกราฟ

```
optimal_clusters = 2

agg_clustering = AgglomerativeClustering(n_clusters=optimal_clusters, linkage='single')

# Fit the model to the cleaned data and obtain cluster labels
cluster_labels = agg_clustering.fit_predict(cleaned_scaled_data)
data_cleaned = data_cleaned.iloc[cleaned_indices].copy()
data_cleaned['cluster'] = cluster_labels

# Analyze the results by examining the distribution of features within each cluster
sns.pairplot(data_cleaned, vars=selected_columns, hue='cluster', diag_kind='hist', corner=True)
plt.show()
```

ภาพที่ ข.21 แสดงผลการจัดกลุ่มในรูปของกราฟโดยเทียบผ่านแต่ละลักษณะที่ได้เลือกมา

```
# Group the data by cluster and genre, then count the data points in each group
genre_counts = data_cleaned.groupby(['cluster', 'genre']).size().reset_index(name='count')

# Rename the columns
genre_counts.columns = ['Cluster', 'Genre', 'Count']

# Sort the data by cluster and count
genre_counts = genre_counts.sort_values(['Cluster', 'Count'], ascending=[True, False])

# Print the genre counts for each cluster
print(genre_counts)
```

ภาพที่ ข.22 แสดงจำนวนข้อมูลการจัดกลุ่มตามลักษณะประเภทเพลง

```

cluster_counts = data_cleaned['cluster'].value_counts().sort_index()
plt.figure(figsize=(10, 5))
ax = cluster_counts.plot(kind='bar')
ax.set_xlabel('Cluster')
ax.set_ylabel('Count')
ax.set_title('Number of Data Points in Each Cluster')
plt.xticks(rotation=0)
plt.show()

```

ภาพที่ ข.23 กราฟแท่งแสดงจำนวนกลุ่มข้อมูลในแต่ละกลุ่ม

```

# Count the number of data points in each cluster
cluster_counts = data_cleaned['cluster'].value_counts()

# Sort the cluster counts by cluster number
cluster_counts = cluster_counts.sort_index()

# Print the cluster counts
print("Number of data points in each cluster:")
print(cluster_counts)

```

ภาพที่ ข.24 แสดงผลจำนวนกลุ่มข้อมูลในแต่ละกลุ่ม

## 8. ตัวโค้ดทั้งหมด

```
import numpy as np

from sklearn.datasets import make_blobs

import matplotlib.pyplot as plt

from sklearn.datasets import make_moons

from sklearn.datasets import make_circles

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette_score

from sklearn import datasets

from scipy.cluster.hierarchy import linkage, dendrogram

from sklearn.cluster import AgglomerativeClustering

from sklearn.preprocessing import StandardScaler

def euclid_distance(p1, p2):

    return np.sqrt(np.sum(np.square(p1 - p2)))

def Neighborhood(arr):

    arr_size = len(arr)

    n_arr = arr.reshape((1, arr_size))

    distance_diff = n_arr - n_arr.T

    nbh = np.sum(np.where(distance_diff >= 0, 1, 0), axis=0)

    return nbh

def NBH_Matrix(data):

    d_size = len(data)
```

```

def NBH_Matrix(data):

    d_size = len(data)

    dist_matrix = np.zeros((d_size, d_size))

    for i in range(d_size):

        for j in range(i+1, d_size):

            dist_matrix[i,j] = euclid_distance(data[i], data[j])

            dist_matrix[j,i] = dist_matrix[i,j]

    Neighbor_matrix = np.ones((d_size, d_size))

    Neighbor_matrix = np.apply_along_axis(Neighborhood, 1, dist_matrix)

    return Neighbor_matrix

def MassRatio(data):

    minor_NBH_matrix = NBH_Matrix(data)

    return minor_NBH_matrix / np.transpose(minor_NBH_matrix)

def MOF_p(pre_arr, i):

    arr = np.delete(pre_arr, i, axis=0)

    return np.var(arr)

def MOF(data):

    MR_Matrix = MassRatio(data)

    d_size = len(data)

    MRV_Matrix = np.zeros(d_size)

    for i in range(d_size):

        MRV_Matrix[i] = MOF_p(MR_Matrix[:, i], i)

    return MRV_Matrix

```

```
def find_outliers(data, percentile=95):

    # Calculate MOF values

    mof_values = MOF(data)

    # Determine the threshold based on the specified percentile

    threshold = np.percentile(mof_values, percentile)

    # Find outliers based on the calculated threshold

    outliers = np.where(mof_values > threshold)

    return outliers

def remove_outliers(data, percentile=95):

    # Find outlier indices using the find_outliers function

    outlier_indices = find_outliers(data, percentile)

    # Remove outliers from the data

    cleaned_data = np.delete(data, outlier_indices, axis=0)

    # Get the indices of the cleaned data in the original dataset

    cleaned_indices = np.setdiff1d(np.arange(data.shape[0]), outlier_indices)

    return cleaned_data, cleaned_indices
```

```
# Plot the original data

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)

plt.scatter(data[:, 0], data[:, 1], c=cluster_assignments_original, cmap='viridis')

plt.scatter(cluster_centroids_original[:, 0], cluster_centroids_original[:, 1], c='red', marker='x', s=100, label='Centroids')

plt.title('K-means Clustering (Original Data)')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend()

# Plot the cleaned data

plt.subplot(1, 2, 2)

plt.scatter(cleaned_data[:, 0], cleaned_data[:, 1], c=cluster_assignments_cleaned, cmap='viridis')

plt.scatter(cluster_centroids_cleaned[:, 0], cluster_centroids_cleaned[:, 1], c='red', marker='x', s=100, label='Centroids')

plt.scatter(data[outlier_indices, 0], data[outlier_indices, 1], c='black', marker='x', s=100, label='Outliers')

plt.title('K-means Clustering (Cleaned Data)')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend()

plt.show()
```

```
# Calculate the silhouette score for the original data

silhouette_score_original = silhouette_score(data, cluster_assignments_original)

# Calculate the silhouette score for the cleaned data

silhouette_score_cleaned = silhouette_score(cleaned_data, cluster_assignments_cleaned)

print("Silhouette score (original data):", silhouette_score_original)

print("Silhouette score (cleaned data):", silhouette_score_cleaned)

# Generate synthetic data with noisy circles

data, labels = make_circles(n_samples=400, factor=0.5, noise=0.05, random_state=42)

# Remove outliers from the dataset

cleaned_data, cleaned_indices = remove_outliers(data, percentile=95)

# Get outlier indices

outlier_indices = np.setdiff1d(np.arange(data.shape[0]), cleaned_indices)

# Determine the optimal number of clusters using the Silhouette Method

max_clusters = 10

optimal_clusters = 2

# Apply k-means clustering to the original data

cluster_assignments_original, cluster_centroids_original = k_means_clustering(data, optimal_clusters)

# Apply k-means clustering to the cleaned data

cluster_assignments_cleaned, cluster_centroids_cleaned = k_means_clustering(cleaned_data, optimal_clusters)
```

```
# Plot the original data

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)

plt.scatter(data[:, 0], data[:, 1], c=cluster_assignments_original, cmap='viridis')

plt.scatter(cluster_centroids_original[:, 0], cluster_centroids_original[:, 1], c='red', marker='x', s=100, label='Centroids')

plt.title('K-means Clustering (Original Data)')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend()

# Plot the cleaned data

plt.subplot(1, 2, 2)

plt.scatter(cleaned_data[:, 0], cleaned_data[:, 1], c=cluster_assignments_cleaned, cmap='viridis')

plt.scatter(cluster_centroids_cleaned[:, 0], cluster_centroids_cleaned[:, 1], c='red', marker='x', s=100, label='Centroids')

plt.scatter(data[outlier_indices, 0], data[outlier_indices, 1], c='black', marker='x', s=100, label='Outliers')

plt.title('K-means Clustering (Cleaned Data)')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend()

plt.show()

# Calculate the silhouette score for the original data

silhouette_score_original = silhouette_score(data, cluster_assignments_original)
```

```

# Calculate the silhouette score for the cleaned data

silhouette_score_cleaned = silhouette_score(cleaned_data, cluster_assignments_cleaned)

print("Silhouette score (original data):", silhouette_score_original)

print("Silhouette score (cleaned data):", silhouette_score_cleaned)

# Generate synthetic data with no specific structure

data, labels = make_blobs(n_samples=400, centers=1, cluster_std=5.0, random_state=42)

cleaned_data, cleaned_indices = remove_outliers(data, percentile=95)

outlier_indices = np.setdiff1d(np.arange(data.shape[0]), cleaned_indices)

max_clusters = 10

optimal_clusters_original = find_optimal_clusters_silhouette(data, max_clusters)

optimal_clusters_cleaned = find_optimal_clusters_silhouette(cleaned_data, max_clusters)

# Apply k-means clustering to the original data

cluster_assignments_original, cluster_centroids_original = k_means_clustering(data, optimal_clusters_original)

# Apply k-means clustering to the cleaned data

cluster_assignments_cleaned, cluster_centroids_cleaned = k_means_clustering(cleaned_data, optimal_clusters_cleaned)

# Plot the original data

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)

plt.scatter(data[:, 0], data[:, 1], c=cluster_assignments_original, cmap='viridis')

plt.scatter(cluster_centroids_original[:, 0], cluster_centroids_original[:, 1], c='red', marker='x', s=100, label='Centroids')

plt.title('K-means Clustering (Original Data)')

```

```
plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend()

# Plot the cleaned data

plt.subplot(1, 2, 2)

plt.scatter(cleaned_data[:, 0], cleaned_data[:, 1], c=cluster_assignments_cleaned, cmap='viridis')

plt.scatter(cluster_centroids_cleaned[:, 0], cluster_centroids_cleaned[:, 1], c='red', marker='x', s=100, label='Centroids')

plt.scatter(data[outlier_indices, 0], data[outlier_indices, 1], c='black', marker='x', s=100, label='Outliers')

plt.title('K-means Clustering (Cleaned Data)')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend()

plt.show()

# Calculate the silhouette score for the original data

if optimal_clusters_original > 1:

    silhouette_score_original = silhouette_score(data, cluster_assignments_original)

else:

    silhouette_score_original = None

# Calculate the silhouette score for the cleaned data

if optimal_clusters_cleaned > 1:

    silhouette_score_cleaned = silhouette_score(cleaned_data, cluster_assignments_cleaned)

else:
```

```

silhouette_score_cleaned = None

# Plot the elbow method for the cleaned data

plot_elbow_method(cleaned_data, max_clusters)

print("Silhouette score (original data):", silhouette_score_original)

print("Silhouette score (cleaned data):", silhouette_score_cleaned)

# Generate synthetic data with noisy moons shape

data, labels = make_moons(n_samples=400, noise=0.1, random_state=42)

# Remove outliers from the dataset

cleaned_data, cleaned_indices = remove_outliers(data, percentile=95)

# Get outlier indices

outlier_indices = np.setdiff1d(np.arange(data.shape[0]), cleaned_indices)

# Determine the optimal number of clusters using the Silhouette Method

max_clusters = 10

optimal_clusters_original = find_optimal_clusters_silhouette(data, max_clusters)

optimal_clusters_cleaned = find_optimal_clusters_silhouette(cleaned_data, max_clusters)

optimal_clusters_original = 2

optimal_clusters_cleaned = 2

# Apply k-means clustering to the original data

cluster_assignments_original, cluster_centroids_original = k_means_clustering(data, optimal_clusters_origin al)

# Apply k-means clustering to the cleaned data

cluster_assignments_cleaned, cluster_centroids_cleaned = k_means_clustering(cleaned_data, optimal_clusters_cleaned)

```

```
# Plot the original data

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)

plt.scatter(data[:, 0], data[:, 1], c=cluster_assignments_original, cmap='viridis')

plt.scatter(cluster_centroids_original[:, 0], cluster_centroids_original[:, 1], c='red', marker='x', s=100, label='Centroids')

plt.title('K-means Clustering (Original Data)')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend()

# Plot the cleaned data

plt.subplot(1, 2, 2)

plt.scatter(cleaned_data[:, 0], cleaned_data[:, 1], c=cluster_assignments_cleaned, cmap='viridis')

plt.scatter(cluster_centroids_cleaned[:, 0], cluster_centroids_cleaned[:, 1], c='red', marker='x', s=100, label='Centroids')

plt.scatter(data[outlier_indices, 0], data[outlier_indices, 1], c='black', marker='x', s=100, label='Outliers')

plt.title('K-means Clustering (Cleaned Data)')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend()

plt.show()

# Calculate the silhouette score for the original data

silhouette_score_original = silhouette_score(data, cluster_assignments_original)
```

```
# Calculate the silhouette score for the cleaned data

silhouette_score_cleaned = silhouette_score(cleaned_data, cluster_assignments_cleaned)

print("Silhouette score (original data):", silhouette_score_original)

print("Silhouette score (cleaned data):", silhouette_score_cleaned)

# Generate synthetic data with blobs shape

data, labels = make_blobs(n_samples=400, centers=3, random_state=42, cluster_std=2.0)

# Apply a transformation to create an anisotropic shape

transformation = np.array([[0.6, -0.6], [-0.4, 0.8]])

data = np.dot(data, transformation)

# Scale the data

scaler = StandardScaler()

data = scaler.fit_transform(data)

# Remove outliers from the dataset

cleaned_data, cleaned_indices = remove_outliers(data, percentile=95)

# Get outlier indices

outlier_indices = np.setdiff1d(np.arange(data.shape[0]), cleaned_indices)

# Determine the optimal number of clusters using the Silhouette Method

optimal_clusters_original = 3

optimal_clusters_cleaned = 3

# Apply k-means clustering to the original data

cluster_assignments_original, cluster_centroids_original = k_means_clustering(data, optimal_clusters_origin al)
```

```
# Apply k-means clustering to the cleaned data

cluster_assignments_cleaned, cluster_centroids_cleaned = k_means_clustering(cleaned_data, optimal_clusters_cleaned)

# Plot the original data

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)

plt.scatter(data[:, 0], data[:, 1], c=cluster_assignments_original, cmap='viridis')

plt.scatter(cluster_centroids_original[:, 0], cluster_centroids_original[:, 1], c='red', marker='x', s=100, label='Centroids')

plt.title('K-means Clustering (Original Data)')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend()

# Plot the cleaned data

plt.subplot(1, 2, 2)

plt.scatter(cleaned_data[:, 0], cleaned_data[:, 1], c=cluster_assignments_cleaned, cmap='viridis')

plt.scatter(cluster_centroids_cleaned[:, 0], cluster_centroids_cleaned[:, 1], c='red', marker='x', s=100, label='Centroids')

plt.scatter(data[outlier_indices, 0], data[outlier_indices, 1], c='black', marker='x', s=100, label='Outliers')

plt.title('K-means Clustering (Cleaned Data)')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend()

plt.show()
```

```

# Calculate the silhouette score for the original data

silhouette_score_original = silhouette_score(data, cluster_assignments_original)

# Calculate the silhouette score for the cleaned data

silhouette_score_cleaned = silhouette_score(cleaned_data, cluster_assignments_cleaned)

print("Silhouette score (original data):", silhouette_score_original)

print("Silhouette score (cleaned data):", silhouette_score_cleaned)

# Generate synthetic data with varied cluster standard deviations

data, labels = make_blobs(n_samples=400, centers=3, cluster_std=[1.0, 2.5, 0.5], random_state=170)

# Remove outliers from the dataset

cleaned_data, cleaned_indices = remove_outliers(data, percentile=95)

# Get outlier indices

outlier_indices = np.setdiff1d(np.arange(data.shape[0]), cleaned_indices)

# Determine the optimal number of clusters using the Silhouette Method

max_clusters = 10

optimal_clusters = find_optimal_clusters_silhouette(cleaned_data, max_clusters)

# Apply k-means clustering to the original data

cluster_assignments_original, cluster_centroids_original = k_means_clustering(data, optimal_clusters)

# Apply k-means clustering to the cleaned data

cluster_assignments_cleaned, cluster_centroids_cleaned = k_means_clustering(cleaned_data, optimal_clusters)

# Plot the original data

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)

```

```
plt.scatter(data[:, 0], data[:, 1], c=cluster_assignments_original, cmap='viridis')

plt.scatter(cluster_centroids_original[:, 0], cluster_centroids_original[:, 1], c='red', marker='x', s=100, label='Centroids')

plt.title('K-means Clustering (Original Data)')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend()

# Plot the cleaned data

plt.subplot(1, 2, 2)

plt.scatter(cleaned_data[:, 0], cleaned_data[:, 1], c=cluster_assignments_cleaned, cmap='viridis')

plt.scatter(cluster_centroids_cleaned[:, 0], cluster_centroids_cleaned[:, 1], c='red', marker='x', s=100, label='Centroids')

plt.scatter(data[outlier_indices, 0], data[outlier_indices, 1], c='black', marker='x', s=100, label='Outliers')

plt.title('K-means Clustering (Cleaned Data)')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend()

plt.show()

# Calculate the silhouette score for the original data

silhouette_score_original = silhouette_score(data, cluster_assignments_original)

# Calculate the silhouette score for the cleaned data

silhouette_score_cleaned = silhouette_score(cleaned_data, cluster_assignments_cleaned)

print("Silhouette score (original data):", silhouette_score_original)

print("Silhouette score (cleaned data):", silhouette_score_cleaned)
```

```
def plot_dendrogram(linkage_matrix, **kwargs):
    # Plot the corresponding dendrogram
    dendrogram(linkage_matrix, **kwargs)

def hierarchical_clustering(data, n_clusters, linkage_method='single', plot_dendro=False):
    # Compute the linkage matrix
    linkage_matrix = linkage(data, method=linkage_method)
    model = AgglomerativeClustering(n_clusters=n_clusters, linkage=linkage_method, distance_threshold=N
                                     one)
    # Fit the model to the data
    model = model.fit(data)
    if plot_dendro:
        plt.figure(figsize=(12, 6))
        plot_dendrogram(linkage_matrix, truncate_mode='level', p=400)
        plt.xlabel("Index of data points")
        plt.ylabel("Euclidean distance")
        plt.show()
    # Return the cluster labels for each data point
    return model.labels_
```

```

# Visualization function

def plot_clusters(data, cleaned_data, cleaned_indices, labels):

    plt.scatter(cleaned_data[:, 0], cleaned_data[:, 1], c=labels, cmap='viridis', s=10)

    outlier_indices = np.setdiff1d(np.arange(data.shape[0]), cleaned_indices)

    plt.scatter(data[outlier_indices, 0], data[outlier_indices, 1], c='red', marker='x', s=50, label='Outliers')

    plt.xlabel('Feature 1')

    plt.ylabel('Feature 2')

    plt.legend()

    plt.show()

n_samples = 400

noisy_circles = datasets.make_circles(n_samples=n_samples, factor=0.5, noise=0.05)

noisy_moons = datasets.make_moons(n_samples=n_samples, noise=0.05)

blobs = datasets.make_blobs(n_samples=n_samples, random_state=8)

no_structure = np.random.rand(n_samples, 2), None

random_state = 170

X, y = datasets.make_blobs(n_samples=n_samples, random_state=random_state)

transformation = [[0.6, -0.6], [-0.4, 0.8]]

X_aniso = np.dot(X, transformation)

aniso = (X_aniso, y)

varied = datasets.make_blobs(
    n_samples=n_samples, cluster_std=[1.0, 2.5, 0.5], random_state=random_state
)

datasets = [noisy_circles, noisy_moons, blobs, no_structure, aniso, varied]

```

```
# Loop through each dataset, perform clustering, and visualize the results

for i, dataset in enumerate(datasets):

    print(f"Dataset {i + 1}:")

    X, y = dataset

    # Set a flag to indicate whether the dataset is either noisy_circles or noisy_moons

    is_noisy_circles_or_moons = i in [0, 1]

    n_clusters = 2 if is_noisy_circles_or_moons else 3

    print("Without eliminating outliers:")

    labels_without_outliers_removed = hierarchical_clustering(X, n_clusters=n_clusters, linkage_method='single', plot_dendro=True)

    plot_clusters(X, X, np.arange(X.shape[0]), labels_without_outliers_removed)

    silhouette_score_without_outliers_removed = silhouette_score(X, labels_without_outliers_removed)

    print(f"Silhouette Score: {silhouette_score_without_outliers_removed:.4f}")

    print("With eliminating outliers:")

    cleaned_data, cleaned_indices = remove_outliers(X, percentile=95)

    labels_with_outliers_removed = hierarchical_clustering(cleaned_data, n_clusters=n_clusters, linkage_m
ethod='single',plot_dendro=True)

    plot_clusters(X, cleaned_data, cleaned_indices, labels_with_outliers_removed)

    silhouette_score_with_outliers_removed = silhouette_score(cleaned_data, labels_with_outliers_remov
ed)

    print(f"Silhouette Score: {silhouette_score_with_outliers_removed:.4f}")
```

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.cluster.hierarchy import dendrogram, linkage

from sklearn.preprocessing import StandardScaler

from google.colab import files

uploaded = files.upload()

dataset_path = "spotify_dataset10.csv"

data = pd.read_csv(dataset_path)

def systematic_sampling(data, n_samples):

    data_size = len(data)

    k = data_size // n_samples

    random_start = np.random.randint(0, k)

    indices = np.arange(random_start, data_size, k)[:n_samples]

    sampled_data = data.iloc[indices]

    # Reset the index and drop the old index column

    sampled_data = sampled_data.reset_index(drop=True)

    return sampled_data

n_samples = 8000

sampled_data = systematic_sampling(data, n_samples)

print(sampled_data)
```

```
sampled_data.info()

sampled_data.head()

# Check for missing values

missing_values = sampled_data.isnull().sum()

missing_values

selected_columns1 = ['acousticness', 'danceability', 'energy', 'instrumentalness', 'liveness', 'loudness', 'speec
hiness', 'tempo', 'valence','duration_ms']

data_selected = sampled_data[selected_columns1]

plt.figure(figsize=(15, 10))

for idx, column in enumerate(selected_columns1):

    plt.subplot(3, 4, idx + 1)

    sns.histplot(data=data_selected, x=column, kde=True, color='blue', bins=20)

    plt.title(f'Histogram of {column}')

plt.tight_layout()

plt.show()

genre_count = sampled_data['genre'].value_counts()

fig = px.pie(names=genre_count.index, values=genre_count.values, title='Pie Chart of Genre Counts')

fig.show()

data_clean = sampled_data.drop_duplicates()

selected_columns = ['danceability', 'energy', 'loudness', 'speechiness', 'acousticness', 'liveness', 'valence', 'te
mpo','instrumentalness']

data_selected = data_clean[selected_columns].values

cleaned_data, cleaned_indices = remove_outliers(data_selected, percentile=95)
```

```
data_clean.describe()

genres_cleaned = data_clean['genre'].iloc[cleaned_indices]

genre_count_cleaned = genres_cleaned.value_counts()

# Create a pie chart of the genre counts in the cleaned data

fig = px.pie(names=genre_count_cleaned.index, values=genre_count_cleaned.values, title='Pie Chart of Ge
nre Counts (Cleaned Data)')

fig.show()

outlier_indices = find_outliers(data_selected)

outliers = data_clean.iloc[outlier_indices]

print(f"Number of outliers: {len(outliers)}")

display(outliers)

outliers.describe()

linked = linkage(cleaned_scaled_data, method='single')

plt.figure(figsize=(12, 6))

dendrogram(linked, truncate_mode='level', p=5)

plt.title('Hierarchical Clustering Dendrogram (Single)')

plt.xlabel('Index')

plt.ylabel('Single Linkage Distance')

plt.show()
```

```
linked = linkage(data_scaled, method='single')

plt.figure(figsize=(12, 6))

dendrogram(linked, truncate_mode='level', p=5)

plt.title('Hierarchical Clustering Dendrogram (Single)')

plt.xlabel('Index')

plt.ylabel('Single Linkage Distance')

plt.show()

scaler = StandardScaler()

data_scaled = scaler.fit_transform(data_selected)

silhouette_scores_original = []

silhouette_scores_cleaned = []

for n_clusters in range(2, 11):

    # Apply AgglomerativeClustering for both cases

    model_original = AgglomerativeClustering(n_clusters=n_clusters, linkage='single')

    model_original.fit(data_scaled)

    cluster_labels_original = model_original.labels_

    model_cleaned = AgglomerativeClustering(n_clusters=n_clusters, linkage='single')

    model_cleaned.fit(cleaned_scaled_data)

    cluster_labels_cleaned = model_cleaned.labels_

    # Calculate silhouette scores for both cases

    silhouette_avg_original = silhouette_score(data_scaled, cluster_labels_original)

    silhouette_avg_cleaned = silhouette_score(cleaned_scaled_data, cluster_labels_cleaned)
```

```
# Append silhouette scores to the respective lists

silhouette_scores_original.append(silhouette_avg_original)

silhouette_scores_cleaned.append(silhouette_avg_cleaned)

# Create a pandas DataFrame to display the results

results_df = pd.DataFrame({

    'Number of Clusters': range(2, 11),

    'Silhouette Score (Original Data)': silhouette_scores_original,

    'Silhouette Score (Cleaned Data)': silhouette_scores_cleaned

})

# Display the results in table form

print(results_df)

cluster_range = range(2, 11)

plt.figure(figsize=(10, 5))

plt.plot(cluster_range, silhouette_scores_original, marker='o', linestyle='--',
        label="Without Outlier Elimination")

plt.plot(cluster_range, silhouette_scores_cleaned, marker='o', linestyle='--',
        label="With Outlier Elimination")

plt.xlabel("Number of clusters")

plt.ylabel("Silhouette Score")

plt.title("Silhouette Scores: With vs. Without Outlier Elimination (Agglomerative Clustering)")

plt.grid()

plt.legend()

plt.show()
```

```
optimal_clusters = 2

agg_clustering = AgglomerativeClustering(n_clusters=optimal_clusters, linkage='single')

# Fit the model to the cleaned data and obtain cluster labels

cluster_labels = agg_clustering.fit_predict(cleaned_scaled_data)

data_cleaned = data_cleaned.iloc[cleaned_indices].copy()

data_cleaned['cluster'] = cluster_labels

# Analyze the results by examining the distribution of features within each cluster

sns.pairplot(data_cleaned, vars=selected_columns, hue='cluster', diag_kind='hist', corner=True)

plt.show()

# Count the number of data points in each cluster

cluster_counts = data_cleaned['cluster'].value_counts()

# Sort the cluster counts by cluster number

cluster_counts = cluster_counts.sort_index()

# Print the cluster counts

print("Number of data points in each cluster:")

print(cluster_counts)

def calculate_wcss(data):

    wcss = []

    cluster_range = range(2, 11)

    for k in cluster_range:

        kmeans = KMeans(n_clusters=k, random_state=42)

        kmeans.fit(data)

        wcss.append(kmeans.inertia_)

return wcss
```

```
wcss_full = calculate_wcss(data_scaled)

wcss_cleaned = calculate_wcss(cleaned_scaled_data)

plt.figure(figsize=(10, 5))

plt.plot(cluster_range, wcss_full, marker='o', linestyle='--', label="Without Outlier Elimination")

plt.plot(cluster_range, wcss_cleaned, marker='o', linestyle='--', label="With Outlier Elimination")

plt.xlabel("Number of clusters")

plt.ylabel("WCSS (Within-Cluster Sum of Squares)")

plt.title("Elbow Plot: With vs. Without Outlier Elimination")

plt.grid()

plt.legend()

plt.show()

def calculate_silhouette_scores(data):

    silhouette_scores = []

    cluster_range = range(2, 11)

    for k in cluster_range:

        kmeans = KMeans(n_clusters=k, random_state=42)

        kmeans.fit(data)

        sil_score = silhouette_score(data, kmeans.labels_)

        silhouette_scores.append(sil_score)

    return silhouette_scores
```

```
silhouette_scores_full = calculate_silhouette_scores(data_scaled)

silhouette_scores_cleaned = calculate_silhouette_scores(cleaned_scaled_data)

plt.figure(figsize=(10, 5))

plt.plot(cluster_range, silhouette_scores_full, marker='o', linestyle='--', label="Without Outlier Elimination")

plt.plot(cluster_range, silhouette_scores_cleaned, marker='o', linestyle='--',
', label="With Outlier Elimination")

plt.xlabel("Number of clusters")

plt.ylabel("Silhouette Score")

plt.title("Silhouette Scores: With vs. Without Outlier Elimination")

plt.grid()

plt.legend()

plt.show()

# Create a pandas DataFrame to display the results

results_df = pd.DataFrame{

    'Number of Clusters': list(cluster_range),

    'Silhouette Score (Without Outlier Elimination)': silhouette_scores_full,

    'Silhouette Score (With Outlier Elimination)': silhouette_scores_cleaned

}

# Display the results in table form

print(results_df)
```

```
optimal_clusters = 2

kmeans_clustering = KMeans(n_clusters=optimal_clusters, random_state=42)

# Fit the model to the cleaned data and obtain cluster labels

cluster_labels = kmeans_clustering.fit_predict(cleaned_scaled_data)

data_cleaned = data_clean.iloc[cleaned_indices].copy()

data_cleaned['cluster'] = cluster_labels

# Analyze the results by examining the distribution of features within each cluster

sns.pairplot(data_cleaned, vars=selected_columns, hue='cluster', diag_kind='hist', corner=True)

plt.show()

def plot_attribute_bars(data, attribute, cluster_col='cluster', bins=5):

    # Create a temporary DataFrame with the attribute and cluster columns

    temp_df = data[[attribute, cluster_col]]

    # Calculate the minimum and maximum values of the attribute

    min_val = temp_df[attribute].min()

    max_val = temp_df[attribute].max()

    # Calculate the bin width

    bin_width = (max_val - min_val) / bins

    # Create bins

    bin_edges = np.arange(min_val, max_val + bin_width, bin_width)

    bin_labels = [f'{round(edge, 2)} - {round(edge + bin_width, 2)}' for edge in bin_edges[:-1]]

    # Assign each data point to a bin

    temp_df['bin'] = pd.cut(temp_df[attribute], bins=bin_edges, labels=bin_labels, include_lowest=True)
```

```
# Count the number of data points in each bin for each cluster

bin_counts = temp_df.groupby([cluster_col, 'bin']).size().unstack().T

# Plot the bar graph

ax = bin_counts.plot(kind='bar', figsize=(10, 5))

ax.set_xlabel(f'{attribute} ranges')

ax.set_ylabel('Count')

ax.set_title(f'{attribute} Distribution by Cluster')

plt.xticks(rotation=45)

plt.legend(title='Cluster', loc='upper right')

plt.show()

optimal_clusters = 2

kmeans_clustering = KMeans(n_clusters=optimal_clusters, random_state=42)

cluster_labels = kmeans_clustering.fit_predict(cleaned_scaled_data)

cleaned_data_original_scale = scaler.inverse_transform(cleaned_scaled_data)

data_cleaned_original_scale = pd.DataFrame(cleaned_data_original_scale, columns=selected_columns, index=cleaned_indices)

data_cleaned_original_scale['cluster'] = cluster_labels

for column in selected_columns:

    plot_attribute_bars(data_cleaned_original_scale, column)
```

```
def print_attribute_bin_counts(data, attribute, cluster_col='cluster', bins=5):

    # Create a temporary DataFrame with the attribute and cluster columns

    temp_df = data[[attribute, cluster_col]]

    # Calculate the minimum and maximum values of the attribute

    min_val = temp_df[attribute].min()

    max_val = temp_df[attribute].max()

    # Calculate the bin width

    bin_width = (max_val - min_val) / bins

    # Create bins

    bin_edges = np.arange(min_val, max_val + bin_width, bin_width)

    bin_labels = [f'{round(edge, 2)} - {round(edge + bin_width, 2)}' for edge in bin_edges[:-1]]

    # Assign each data point to a bin

    temp_df['bin'] = pd.cut(temp_df[attribute], bins=bin_edges, labels=bin_labels, include_lowest=True)

    # Count the number of data points in each bin for each cluster

    bin_counts = temp_df.groupby([cluster_col, 'bin']).size().unstack().T

    # Print the bin counts for each attribute, sorted by cluster

    print(f'{attribute} Distribution by Cluster:')

    print(bin_counts)

    print()

    for column in selected_columns:

        print_attribute_bin_counts(data_cleaned_original_scale, column)
```

```
def plot_genre_bars(data, genre_col='genre', cluster_col='cluster'):

    # Create a temporary DataFrame with the genre and cluster columns

    temp_df = data[[genre_col, cluster_col]]

    # Count the number of data points in each genre for each cluster

    genre_counts = temp_df.groupby([cluster_col, genre_col]).size().unstack().T

    # Plot the bar graph

    ax = genre_counts.plot(kind='bar', figsize=(15, 7))

    ax.set_xlabel('Genres')

    ax.set_ylabel('Count')

    ax.set_title('Genre Distribution by Cluster')

    plt.xticks(rotation=45)

    plt.legend(title='Cluster', loc='upper right')

    plt.show()

plot_genre_bars(data_cleaned)

# Group the data by cluster and genre, then count the data points in each group

genre_counts = data_cleaned.groupby(['cluster', 'genre']).size().reset_index(name='count')

# Rename the columns

genre_counts.columns = ['Cluster', 'Genre', 'Count']

# Sort the data by cluster and count

genre_counts = genre_counts.sort_values(['Cluster', 'Count'], ascending=[True, False])

# Print the genre counts for each cluster

print(genre_counts)
```

```
cluster_counts = data_cleaned['cluster'].value_counts().sort_index()

plt.figure(figsize=(10, 5))

ax = cluster_counts.plot(kind='bar')

ax.set_xlabel('Cluster')

ax.set_ylabel('Count')

ax.set_title('Number of Data Points in Each Cluster')

plt.xticks(rotation=0)

plt.show()

# Count the number of data points in each cluster

cluster_counts = data_cleaned['cluster'].value_counts()

# Sort the cluster counts by cluster number

cluster_counts = cluster_counts.sort_index()

# Print the cluster counts

print("Number of data points in each cluster:")

print(cluster_counts)
```

## ประวัติผู้จัดทำโครงการ

นายจิรภัทร ชัยบุตร  
เลขประจำตัวนิสิต 6234307323  
สาขา คณิตศาสตร์  
ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์  
จุฬาลงกรณ์มหาวิทยาลัย

