การจำแนกประเภทของโรคเวียนศีรษะขณะเปลี่ยนท่าจากข้อมูลแบบประเมินอาการวิงเวียนศีรษะโดยใช้เทคนิคการเรียนรู้ของเครื่อง

Classification of Benign Paroxysmal Positioning Vertigo Types from Dizziness Handicap Inventory using Machine Learning Techniques

ลวณา มสารกรัณฑ์

บัณฑิตวิทยาลัยมหาวิทยาลัยศรีนครินทรวิโรฒ

2561

การจำแนกประเภทของโรคเวียนศีรษะขณะเปลี่ยนท่าจากข้อมูลแบบประเมินอาการวิงเวียนศีรษะโดยใช้เทคนิคการเรียนรู้ของเครื่อง

ลวณา มสารกรัณฑ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ
ปีการศึกษา 2561

Classification of Benign Paroxysmal Positioning Vertigo Types from Dizziness Handicap Inventory using Machine Learning Techniques

LAWANA MASANKARAN

A Thesis Submitted in partial Fulfillment of Requirements

for MASTER OF SCIENCE (Information Technology)

Faculty of Science Srinakharinwirot University

2018

ปริญญานิพนธ์

เรื่อง

การจำแนกประเภทของโรคเวียนศีรษะขณะเปลี่ยนท่าจากข้อมูลแบบประเมินอาการวิงเวียนศีรษะโดยใช้เทคนิคการเรียนรู้ของเครื่อง

ของ

ลวณา มสารกรัณฑ์

ได้รับอนุมัติจากบัณฑิตวิทยาลัยให้นับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
ของมหาวิทยาลัยศรีนครินทรวิโรฒ

..... คณบดีบัณฑิตวิทยาลัย
(ผู้ช่วยศาสตราจารย์ นายแพทย์ฉัตรชัย เอกปัญญาสกุล)

.....

คณะกรรมการสอบปากเปล่าปริญญานิพนธ์

..... ที่ปรึกษาหลัก
(ผู้ช่วยศาสตราจารย์ ดร.วราภรณ์ วิยานนท์)

..... ประธาน
(ผู้ช่วยศาสตราจารย์ ดร.อัครา ประโยชน์)

..... ที่ปรึกษาร่วม
(ผู้ช่วยศาสตราจารย์ ดร.วิศาล มหาสิทธิวัฒน์)

..... กรรมการ
(อาจารย์ ดร.ศิริสรรพ เหล่าหะเกียรติ)

ชื่อเรื่อง	การจำแนกประเภทของโรคเวียนศีรษะขณะเปลี่ยนท่าจากข้อมูลแบบประเมินอาการวิงเวียนศีรษะโดยใช้เทคนิคการเรียนรู้ของเครื่อง
ผู้วิจัย	ลวณา มสารกรัณฑ์
ปริญญา	วิทยาศาสตรมหาบัณฑิต
ปีการศึกษา	2561
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร. วราภรณ์ วิยานนท์

โรคเวียนศีรษะขณะเปลี่ยนท่าเป็นหนึ่งในสาเหตุหลักของอาการเวียนศีรษะซึ่งส่งผลกระทบต่อการใช้ชีวิตประจำวันของผู้ป่วยอย่างมาก ประเภทที่ต่างกันของโรคเวียนศีรษะขณะเปลี่ยนท่าจะมีการรักษาด้วยการกายภาพบำบัดที่ต่างกัน ซึ่งการแยกประเภทของโรคเวียนศีรษะขณะเปลี่ยนท่าสามารถแยกด้วยลักษณะจำเพาะของทิศทางการสั่นของลูกตา และทำได้โดยแพทย์ผู้เชี่ยวชาญเท่านั้น ดังนั้นการรักษาโรคเวียนศีรษะขณะเปลี่ยนท่าให้ถูกต้องตามประเภทของโรคอาจมีความล่าช้า ผู้ป่วยโรคเวียนศีรษะขณะเปลี่ยนท่าจะได้รับการทดสอบด้วยแบบประเมินอาการวิงเวียนศีรษะก่อนพบแพทย์ เพื่อประเมินความรุนแรงของอาการเวียนศีรษะ หากแบบประเมินอาการวิงเวียนศีรษะสามารถช่วยแพทย์แยกประเภทโรคเวียนศีรษะขณะเปลี่ยนท่าได้จะทำให้การรักษาโรคมีความรวดเร็วและลดผลกระทบต่อผู้ป่วยได้ ในการวิจัยนี้จึงมุ่งศึกษาความสามารถในการแยกประเภทของโรคเวียนศีรษะขณะเปลี่ยนท่าจากแบบประเมินอาการวิงเวียนศีรษะโดยใช้เทคนิคการเรียนรู้ของเครื่องเพื่อสร้างแบบจำลองการทำนายประเภทของโรคเวียนศีรษะขณะเปลี่ยนท่าด้วยอัลกอริธึม Random Forest, Support Vector Machine, K-Nearest Neighbor และ Naive Bayes โดยใช้เทคนิคในการจัดการกับฟีเจอร์ต่างกัน คือ การคัดเลือกฟีเจอร์โดยใช้ Chi-square การคัดเลือกฟีเจอร์ด้วยเทคนิค Recursive Feature Elimination และการสกัดฟีเจอร์ด้วย Principal Component Analysis โดยวัดประสิทธิภาพของแบบจำลองต่าง ๆ ด้วย Accuracy score F1-Score Recall-score และ Precision score โดยแบบจำลองที่ให้ประสิทธิภาพสูงสุด คือ แบบจำลองที่สร้างด้วยอัลกอริธึม Naive Bayes สร้างด้วยฟีเจอร์ F7_E23 E15 E23 F7 P8 และ F24 ในการสร้างแบบจำลอง โดยให้ประสิทธิภาพ ดังนี้ accuracy score 67.51% precision score 67.20% recall score 65.97% และ F1-score 65.95% ซึ่งแบบจำลองที่ได้ยังมีประสิทธิภาพไม่เพียงพอจะใช้ในทางการแพทย์ จึงยังต้องใช้ควบคู่กับการวินิจฉัยของแพทย์เพื่อเก็บข้อมูลต่อ ซึ่งในอนาคตหากมีข้อมูลมากพอ อาจจะสามารถพัฒนาแบบจำลองให้มีประสิทธิภาพเพิ่มขึ้นได้

คำสำคัญ : โรคเวียนศีรษะขณะเปลี่ยนท่า, แบบประเมินอาการวิงเวียนศีรษะ, เทคนิคการเรียนรู้ของเครื่อง

Title	Classification of Benign Paroxysmal Positioning Vertigo Types from Dizziness Handicap Inventory using Machine Learning Techniques
Author	LAWANA MASANKARAN
Degree	MASTER OF SCIENCE
Academic Year	2018
Thesis Advisor	Assistant Professor Doctor WARAPORN VIYANON

Benign Paroxysmal Positioning Vertigo (BPPV) is one of the causes of vertigo which has an extreme effect on the daily lives of patients. the different types of BPPV are also treated differently. For example, physicians differentiate between BPPV types using nystagmus characteristics. However, some patients have an unclear nystagmus, so their treatments are delayed due to the difficulty of establishing a diagnosis. The Dizziness Handicap Inventory (DHI) is a tool used to assess the severity of dizziness before a patient is diagnosed by a physician. The use of DHI can distinguish BPPV types which can help physicians to decide which treatments would be the most beneficial for the patients. This research aims to study the ability to use DHI for differential diagnoses of the posterior canal - Benign Paroxysmal Positioning Vertigo (PC-BPPV) and horizontal canal - Benign Paroxysmal Positioning Vertigo (HC-BPPV) via machine learning techniques. The Random Forest, Support vector machine, K-Nearest Neighbor, and Naïve Bayes were used to develop predictive models for DHI features that had statistical significance. Then, the Chi-square, Recursive feature elimination, and Principal component analysis were used to improve the performance of the models. Furthermore, accuracy, precision, recall, and F1-scores were used to evaluate the performance of each model. It revealed that the features F7_E23, E15, E23, F7, P8, and F24 were the top six important features. The model using the Gaussian Naïve Bayes was the best model for discriminating between HC-BPPV and PC-BPPV with 67.51% accuracy, 67.20% precision, 65.97% recall, and 65.95% F1-score. In conclusion, the models created from the DHI score can predict BPPV types at a certain level which can be used for initial diagnosis, but are not accurate enough for medical treatment. Physicians must use the medical history of the patient and nystagmus observation in order to make a diagnosis. In the future, if more data or features can be collected, it may reduce the overfitting problem and lead to a better performance model.

Keyword : BPPV, DHI, machine learning, vertigo

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้สำเร็จลุล่วงได้ด้วยความอนุเคราะห์จาก ผศ.ดร.วราภรณ์ วิยานนท์ อาจารย์ที่ปรึกษา และ ผศ.นพ.วิศาล มหาสิทธิวัฒน์ อาจารย์ที่ปรึกษาร่วม ที่ให้คำปรึกษา คำแนะนำในการทำปริญญานิพนธ์ ตลอดจนสนับสนุนข้อมูลทางวิชาการ และข้อมูลสำหรับทำปริญญานิพนธ์นี้

ขอกราบขอบพระคุณคณะกรรมการสอบปริญญานิพนธ์ ที่ได้ให้คำแนะนำและข้อเสนอแนะสำหรับการปรับปรุงปริญญานิพนธ์ และการใช้เทคนิคการเรียนรู้ของเครื่องเป็นเครื่องมือในการวิเคราะห์และแก้ไขปัญหาทางข้อมูลต่อไป

ขอกราบขอบพระคุณ บัณฑิตวิทยาลัย มหาวิทยาลัยศรีนครินทรวิโรฒ สำหรับทุนสนับสนุนการนำเสนอผลงานวิจัยของนิสิตบัณฑิตศึกษา ทำให้ได้ประสบการณ์นำเสนอปริญญานิพนธ์ และได้แลกเปลี่ยนความรู้ทางด้านเทคโนโลยีทางการแพทย์ และการใช้เทคนิคการเรียนรู้ของเครื่องทางการแพทย์

ลวณา มสารกรัณฑ์

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญรูปภาพ

หน้า

บทที่ 1
บทนำ

1.1 ภูมิหลัง

การเวียนศีรษะเป็นอาการที่พบได้บ่อยและพบได้ในทุกช่วงอายุ ซึ่งส่งผลกระทบต่อการดำเนินชีวิตประจำวันของผู้ป่วยเป็นอย่างมาก การเวียนศีรษะเกิดจากการที่ระบบการทรงตัวของร่างกายสูญเสียไปชั่วขณะ ทำให้รู้สึกปวดศีรษะ สิ่งรอบตัวหมุน และคลื่นไส้อาเจียน ซึ่งการเวียนศีรษะเกิดได้เนื่องจากหลายสาเหตุและหลายโรค เช่น โรคเวียนศีรษะขณะเปลี่ยนท่า โรคน้ำในหูไม่เท่ากัน สมองกระทบกระเทือน ไมเกรน ความผิดปกติของตา หรือมะเร็ง เป็นต้น ทำให้ยากต่อการวินิจฉัยและหาสาเหตุ การตรวจรักษาอาการเวียนศีรษะจำเป็นต้องมีการวัดระดับการเวียนศีรษะของผู้ป่วย โดยใช้แบบประเมินอาการวิงเวียนศีรษะ หรือ Dizziness Handicap Inventory (DHI) ซึ่งเป็นเครื่องมือประเมินความรุนแรงของอาการเวียนศีรษะเบื้องต้น ร่วมกับการซักประวัติผู้ป่วย

สาเหตุอันดับหนึ่ง หรือประมาณ 16 เปอร์เซ็นต์ของการเวียนศีรษะ [1] เกิดจากโรคเวียนศีรษะขณะเปลี่ยนท่า หรือ โรค Benign Paroxysmal Positioning Vertigo (BPPV) โรค BPPV เกิดจากการที่ตะกอนหินปูนในส่วนยูตริเคิล (Utricle) ซึ่งอยู่ในหูชั้นในหลุดออกมาในส่วนเซมิเซอร์คิวลาร์แคแนล (semicircular canal) ซึ่งเป็นส่วนหนึ่งที่ใช้ในการควบคุมการทรงตัว จึงไปกระตุ้นให้เกิดอาการเวียนศีรษะขึ้น ในปัจจุบันวงการทางการแพทย์ใช้ DHI มาช่วยวินิจฉัยจำแนกอาการเวียนศีรษะที่เกิดจากโรค BPPV จากอาการเวียนศีรษะที่เกิดจากสาเหตุอื่นที่ไม่ใช่ BPPV ได้อย่างมีประสิทธิภาพ [2] แต่โรค BPPV ยังสามารถแบ่งออกได้เป็น 3 ชนิด ตามตำแหน่งที่ตะกอนหินปูนเคลื่อนไปอยู่ คือ 1) โรคเวียนศีรษะขณะเปลี่ยนท่าชนิดโพสทีเรียร์แคแนล (Posterior canal - Benign Paroxysmal Positioning Vertigo: PC-BPPV) คือ ตะกอนหินปูนในหูชั้นในหลุดไปที่ semicircular canal ส่วน Posterior canal 2) โรคเวียนศีรษะขณะเปลี่ยนท่าชนิดฮอริซอนทอลแคแนล (Horizontal canal - Benign Paroxysmal Positioning Vertigo: HC-BPPV) คือ ตะกอนหินปูนในหูชั้นในหลุดไปที่ semicircular canal ส่วน Horizontal canal 3) โรคเวียนศีรษะขณะเปลี่ยนท่าชนิดแอนทีเรียร์แคแนล (Anterior canal - Benign Paroxysmal Positioning Vertigo: AC-BPPV) คือ ตะกอนหินปูนในหูชั้นในหลุดไปที่ semicircular canal ส่วน Anterior canal แต่โรคเวียนศีรษะขณะเปลี่ยนท่าชนิดแอนทีเรียร์แคแนลพบได้น้อยมาก การรักษาด้วยการทำกายภาพบำบัดสำหรับโรค BPPV แต่ละแบบมีวิธีการที่แตกต่างกัน และต้องใช้เวลาในการวินิจฉัย หากนำเทคนิคการเรียนรู้ของเครื่อง (Machine learning) มาช่วยในการวินิจฉัยจำแนกชนิดของ BPPV ด้วยแบบประเมิน DHI จะทำให้

สามารถลดเวลาการวินิจฉัย จึงมีประโยชน์และมีความสำคัญต่อการรักษาผู้ป่วย และช่วยบรรเทาผลกระทบของอาการเวียนศีรษะในชีวิตประจำวันของผู้ป่วยได้

1.2 ความมุ่งหมายของงานวิจัย

ในการวิจัยครั้งนี้ผู้วิจัยได้ตั้งความมุ่งหมายไว้ดังนี้

1.2.1 เพื่อสร้างแบบจำลองเพื่อช่วยทำนายการวินิจฉัยจำแนกโรค HC-BPPV และ PC-BPPV โดยใช้เทคนิค Machine learning กับข้อมูล DHI ในผู้ป่วย BPPV

1.2.2 เพื่อศึกษาว่าตัวแปรใดในแบบทดสอบ DHI ที่มีความสำคัญในการทำนายโรคโดยสร้างโมเดลที่ใช้ตัวแปรต่างกันมาเปรียบเทียบ

1.2.3 เพื่อหาความแม่นยำของแต่ละอัลกอริธึมที่แตกต่างกันในการสร้างแบบจำลองเพื่อใช้ข้อมูล DHI ช่วยทำนายการวินิจฉัยจำแนกโรค HC-BPPV และ PC-BPPV

1.2.4 เพื่อมีความรู้ความเข้าใจเกี่ยวกับโรค Benign Paroxysmal Positioning Vertigo (BPPV)

1.2.5 ทราบถึงหลักการของอัลกอริธึม Random Forest, Gaussian Naïve Bayes, K-Nearest Neighbors Support Vector Machine และเครื่องมือ Grid Search และสามารถนำมาใช้งานได้อย่างเหมาะสม

1.2.6 พัฒนาทักษะการสร้างแบบจำลองเพื่อแก้ปัญหาประเภท Classification โดยใช้ Machine Learning

1.2.7 เพื่อศึกษาการทำ Feature Engineering และ Feature Selection ทั้งแบบ Filter method และ Wrapper method ในการคัดเลือกฟีเจอร์

1.2.8 ทราบถึงความแตกต่างของผลคะแนนจากแบบทดสอบ Dizziness Handicap Inventory ที่มีต่อโรค BPPV ชนิด PC-BPPV และ BPPV ชนิด HC-BPPV และสามารถนำไปใช้ประโยชน์ในการสร้างเครื่องมือวินิจฉัยโรค PC-BPPV และ HC-BPPV เพื่อช่วยให้การวินิจฉัยโรคมีความรวดเร็วขึ้นได้

1.3 ความสำคัญของการวิจัย

งานวิจัยนี้ศึกษาการจำแนกโรค Benign Paroxysmal Positioning Vertigo (BPPV) แบบ Posterior canal - Benign Paroxysmal Positioning Vertigo (PC-BPPV) และ Horizontal canal - Benign Paroxysmal Positioning Vertigo (HC-BPPV) โดยใช้ Machine learning เป็นเครื่องมือสำหรับสร้างแบบจำลองในการวินิจฉัยจำแนกโรค ข้อมูลที่ใช้เป็นแบบทดสอบ Dizziness Handicap Inventory (DHI) ที่ใช้เพื่อประเมินผลกระทบของการเวียนศีรษะต่อการดำเนิน

ชีวิตประจำวัน ประกอบด้วยคำถาม 3 ด้าน คือ 1) ด้านอารมณ์ มีคำถาม 9 ข้อ 2) ด้านการเกิดอาการ มีคำถาม 7 ข้อ และ 3) ด้านความสามารถในการใช้ชีวิต มีคำถาม 9 ข้อ รวมมีคำถามทั้งหมด 25 ข้อ

1.4 ขอบเขตของการวิจัย

1.4.1 ประชากรที่ใช้ในการวิจัย

ผู้ป่วยคนไทย 114 คน ในปี 2560 ที่มีอาการเวียนศีรษะจากโรค BPPV ซึ่งได้รับการวินิจฉัยจำแนกโรคจากแพทย์เฉพาะทาง ได้รับการรักษาและติดตามอาการจนหายจากอาการเวียนศีรษะ

1.4.2 กลุ่มตัวอย่างที่ใช้ในการวิจัย

ข้อมูลแบบทดสอบ Dizziness Handicap Inventory (DHI) และอายุของประชากรที่ใช้ในการวิจัย โดยแบ่งเป็น 2 กลุ่มคือ DHI ของผู้ป่วย PC-BPPV และ HC-BPPV

1.4.3 ตัวแปรที่ศึกษา

1.4.3.1 ตัวแปรอิสระ แบ่งเป็นดังนี้

1.4.3.1.1 ข้อมูลผู้ป่วย

- อายุผู้ป่วย

1.4.3.1.2 ข้อมูลจาก DHI

- ข้อมูลจากคำถามประเมินด้านอาการ
  - P1 : เมื่อฉันเงยหน้าขึ้นจะทำให้ฉันมีอาการเวียนหัวมากขึ้น
  - P4 : เมื่อฉันเดินผ่านช่องทางแคบ ๆ เช่นระหว่างชั้นวางของในร้านค้า มักทำให้เวียนหัวมากขึ้น
  - P8 : การทำกิจกรรมที่ใช้แรง เช่น ออกกำลังกาย เต้น การทำงานบ้าน จะยิ่งทำให้ฉันเวียนหัวมากขึ้น
  - P11 : การขยับศีรษะเร็ว ๆ ทำให้ฉันมีอาการเวียนหัวมากขึ้น
  - P13 : เมื่อฉันนอนพลิกตัวบนเตียง ทำให้ฉันมีอาการเวียนหัว มากขึ้น
  - P17 : เวลาที่ฉันเดินไปตามทางเท้า ฉันจะมีอาการเวียนหัวมากขึ้น
  - P25 : การก้มหน้าโค้งลงทำให้ฉันมีอาการเวียนหัวมากขึ้น

- ข้อมูลจากคำถามประเมินด้านอารมณ์
    - E2 : อาการเวียนหัวที่เป็นอยู่ทำให้ฉันรู้สึกสับสนและเป็นกังวล
    - E9 : อาการเวียนหัวที่เป็นอยู่ ทำให้ฉันกลัวที่จะออกจากบ้านเพียงลำพังโดยไม่มีใครไปด้วย
    - E10 : จากอาการเวียนหัวที่ฉันเป็น ทำให้ฉันลำบากใจในการพบปะหรืออยู่ต่อหน้าผู้อื่น
    - E15 : จากที่ฉันมีอาการเวียนหัว ฉันจึงกลัวว่าคนอื่นอาจจะมองว่าฉันเป็นคนขี้เมาหรือติดยาได้
    - E18 : อาการเวียนหัวที่ฉันเป็น ทำให้ฉันรู้สึกยากลำบากที่จะมีสมาธิจดจ่อกับเรื่องใด ๆ ได้
    - E20 : อาการเวียนหัวที่เป็นอยู่ ส่งผลให้ฉันกลัวที่จะอยู่บ้านเพียงลำพังคนเดียว
    - E21 : จากที่ฉันมีอาการเวียนหัว ทำให้ฉันรู้สึกเหมือนว่าฉันเป็นคนพิการ ช่วยตนเองไม่ค่อยได้
    - E22 : อาการเวียนหัว ทำให้ฉันรู้สึกเครียดเกรงว่าสัมพันธภาพในครอบครัวหรือกับเพื่อนจะแย่ลง
    - E23 : อาการเวียนหัวที่เป็นอยู่ ทำให้ฉันรู้สึกหดหู่ใจ ซึมเศร้า
- ข้อมูลจากคำถามประเมินความสามารถในการใช้ชีวิตประจำวัน
    - F3 : อาการเวียนหัวที่เป็นอยู่ ทำให้ฉันมีข้อจำกัดในการเดินทางไปทำธุระหรือพักผ่อนนอกบ้าน
    - F5 : อาการเวียนหัวที่เป็นอยู่ ทำให้ฉันยากลำบากที่จะล้มตัวลงนอนหรือลุกขึ้นจากเตียงทันที
    - F6 : อาการเวียนหัวที่เป็นอยู่ ทำให้ฉันมีข้อจำกัดในการเข้าร่วมงานสังคมนอกบ้าน เช่น งานเลี้ยงงานสังสรรค์ ไปร่วมงานบ้านเพื่อน เป็นต้น

- F7 : อาการเวียนหัวที่เป็นอยู่ ทำให้ฉันยากลำบากมากขึ้นในการอ่านหนังสือ เอกสารต่าง ๆ
- F12 : อาการเวียนหัวที่ฉันเป็นอยู่นี้ ทำให้ฉันต้องหลีกเลี่ยงการขึ้นที่สูง
- F14 : อาการเวียนหัวที่ฉันเป็น ทำให้ฉันยากที่จะทำงานบ้านหรือบริเวณรอบบ้านได้อย่างขันแข็ง
- F16 : จากอาการที่ฉันเวียนหัว ทำให้ฉันยากลำบากที่จะเดินด้วยตนเองโดยไม่พึ่งอุปกรณ์ช่วย
- F19 : อาการเวียนหัวนี้ ทำให้ฉันไม่สามารถเดินในท่ามกลางที่มืด ๆ ได้ แม้อยู่ในบริเวณบ้านก็ตาม
- F24 : อาการเวียนหัวที่ฉันเป็นอยู่นั้นรบกวนการทำงานบ้านหรืองานประจำของฉัน

1.4.3.2 ตัวแปรตาม ได้แก่ ประเภทของ BPPV ซึ่งในการวิจัยนี้มี 2 ประเภท คือ PC-BPPV และ HC-BPPV

1.5 กรอบแนวคิดในงานวิจัย

งานวิจัยนี้ศึกษาการจำแนกโรค Benign Paroxysmal Positioning Vertigo (BPPV) แบบ Posterior canal - Benign Paroxysmal Positioning Vertigo (PC-BPPV) และ Horizontal canal - Benign Paroxysmal Positioning Vertigo (HC-BPPV) โดยใช้ Machine learning เป็นเครื่องมือสำหรับสร้างแบบจำลองในการวินิจฉัยจำแนกโรค ข้อมูลที่ใช้เป็นแบบทดสอบ Dizziness Handicap Inventory (DHI) สำหรับประเมินผลกระทบของการเวียนศีรษะต่อการดำเนินชีวิตประจำวัน ประกอบด้วยคำถาม 3 ด้าน คือ 1) ด้านอารมณ์ จำนวน 9 ข้อ 2) ด้านการเกิดอาการ จำนวน 7 ข้อ และ 3) ด้านความสามารถในการใช้ชีวิต จำนวน 9 ข้อ รวมมีคำถามทั้งหมด 25 ข้อ โดยข้อมูลจะถูกเก็บในรูปแบบ DataFrame จากนั้นจึงใช้ Machine learning มาเป็นเครื่องมือช่วยสร้างแบบจำลองเพื่อทำนายจำแนกชนิดของโรค BPPV ซึ่งเขียนด้วยภาษาโปรแกรม Python โดยใช้เทคนิคการจัดการฟีเจอร์ 2 แบบ คือ การคัดเลือกฟีเจอร์ (Feature Selection) และ การสกัดฟีเจอร์ (Feature Extraction) และใช้เทคนิค Random forest, K-Nearest Neighbors, Naïve Bayes และ Support vector machine ในการแก้ปัญหาประเภท Classification การประเมิน

ประสิทธิภาพจะใช้ค่า Precision, Accuracy, Recall และ F1-score เพื่อเปรียบเทียบความสามารถในการจำแนกโรคของแต่ละแบบจำลองที่สร้างจากอัลกอริธึมที่ต่างกัน เพื่อหาแบบจำลองที่มีประสิทธิภาพในการทำนายสูงที่สุด

1.6 สมมุติฐานในการวิจัย

1.6.1 แบบจำลองที่สร้างด้วยอัลกอริธึม Random Forest มีประสิทธิภาพดีที่สุด

1.6.2 ฟีเจอร์ F7_E23 ทำให้ประสิทธิภาพของการทำนายโดยใช้แบบจำลองดีขึ้น

1.6.3 การใช้ Feature Selection ก่อนสร้างแบบจำลองทำให้แบบจำลองที่มีประสิทธิภาพดีกว่าการสร้างแบบจำลองที่ไม่มีขั้นตอน Feature Selection

บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

ในการวิจัยครั้งนี้ ผู้วิจัยได้ศึกษาเอกสารและงานวิจัยที่เกี่ยวข้อง และได้นำเสนอตามหัวข้อต่อไปนี้

1. โรคเวียนศีรษะขณะเปลี่ยนท่า (Benign Paroxysmal Positioning Vertigo: BPPV)
2. แบบประเมินอาการวิงเวียนศีรษะ (Dizziness Handicap Inventory: DHI)
3. เทคนิคการเรียนรู้ของเครื่อง (Machine Learning)
4. อัลกอริธึมที่ใช้ในเทคนิคการเรียนรู้ของเครื่อง
5. การแบ่งข้อมูลสำหรับทดสอบแบบจำลอง
6. การคัดเลือกฟีเจอร์ (Feature Selection)
7. การสกัดฟีเจอร์ (Feature Extraction)
8. งานวิจัยที่เกี่ยวข้อง

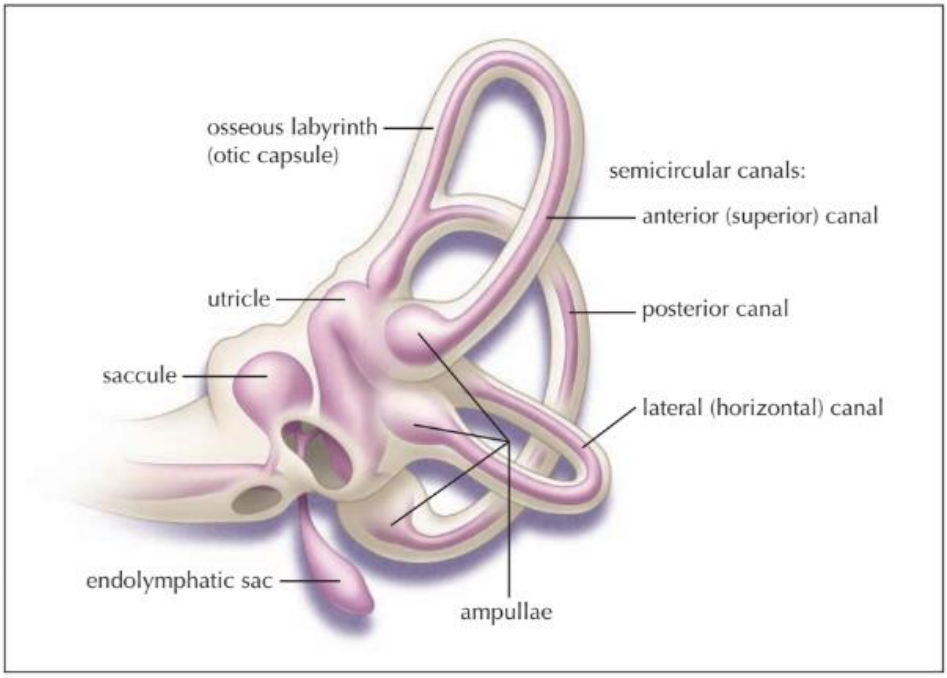
2.1 โรคเวียนศีรษะขณะเปลี่ยนท่า (Benign Paroxysmal Positioning Vertigo: BPPV)

โรค BPPV เป็นหนึ่งในสาเหตุหลักของผู้ป่วยที่มาพบแพทย์ด้วยอาการเวียนศีรษะ โดยโรค BPPV เกิดจากตะกอนหินปูน (otolith) ที่เกาะกันเป็นแผ่นอยู่ที่ Utricle บริเวณหูชั้นใน หลุดออกมาและเคลื่อนตัวไปยังบริเวณ semicircular canal ซึ่งเป็นส่วนหนึ่งในหูที่ทำหน้าที่เกี่ยวกับการทรงตัว และเมื่อผู้ป่วยมีการเคลื่อนไหวร่างกาย ตะกอนหินปูนส่วนที่หลุดอยู่ใน semicircular canal จะเคลื่อนที่ ทำให้ของเหลวเอ็นโดลิมฟ์ (endolymph) ใน semicircular canal เกิดการเคลื่อนไหวและกระตุ้นให้ผู้ป่วยมีอาการเวียนศีรษะได้

ส่วน Semicircular canal ในหูชั้นใน ประกอบด้วยท่อ 3 ชิ้น คือ anterior (superior) canal, posterior canal และ lateral (horizontal) canal ตามภาพประกอบ 1 ซึ่งทำให้ตะกอนหินปูนที่หลุดจาก Utricle มีโอกาสที่จะไปลอยอยู่ใน Semicircular canal ได้ 3 ที่ จึงทำให้ BPPV มี 3 ชนิดด้วยกันขึ้นอยู่กับตำแหน่งที่ตะกอนหินปูนไปลอยอยู่ คือ Posterior canal - Benign Paroxysmal Positioning Vertigo (PC-BPPV), Horizontal canal - Benign Paroxysmal Positioning Vertigo (HC-BPPV) และ Anterior canal - Benign Paroxysmal Positioning Vertigo (AC-BPPV) โดย AC-BPPV พบน้อยมาก ประมาณร้อยละ 2.2 ของผู้ป่วย BPPV [3]

BPPV เป็นโรคที่สามารถหายเองได้แม้ว่าไม่ได้รับการรักษา โดย HC-BPPV จะหายเองภายใน 16±19 วัน และ PC-BPPV จะหายภายใน 39±47 วัน [4] แต่เนื่องจากอาการเวียนหัวเป็นสิ่งที่รบกวนต่อชีวิตประจำวันของผู้ป่วยมาก ดังนั้นหากมีการวินิจฉัยและได้รับการรักษาที่ถูกต้องก็จะช่วยให้ผู้ป่วยสามารถหายจากโรค BPPV ได้เร็วยิ่งขึ้น แพทย์อาจพิจารณาให้ยาเพื่อช่วยลดอาการเวียนหัว

หรืออาการข้างเคียงอื่น แต่การรักษาที่มีประสิทธิภาพที่สุดสำหรับโรค BPPV คือการทำกายภาพบำบัด ซึ่งการทำกายภาพบำบัดในการรักษาโรค BPPV มีจุดมุ่งหมายเพื่อให้ตะกอนหินปูนเคลื่อนออกมาจาก semicircular canal ตำแหน่งที่ต่างกันที่ตะกอนหินปูนหลุดไปอยู่ จึงทำให้ต้องใช้ท่าทางในการกายภาพบำบัดต่างกันด้วย PC-BPPV สามารถรักษาด้วยวิธีที่เรียกว่า Epley's maneuver ในขณะที่ HC-BPPV รักษาด้วยวิธีการ Vannucchi's forced prolonged position [5] ดังนั้นจึงต้องมีการวินิจฉัยเพื่อจำแนกชนิดของ BPPV เพื่อให้การรักษารวดเร็วยิ่งขึ้น



ภาพประกอบ 1 แสดงส่วนประกอบของหูชั้นใน

ที่มา : S. K. A. J. A. Lorne S. Parnes, "Diagnosis and management of benign paroxysmal positional vertigo (BPPV)," *CMAJ,* เล่มที่ 169, %17, pp. 681-693, 2003.

2.2 แบบประเมินอาการวิงเวียนศีรษะ (Dizziness Handicap Inventory: DHI)

แบบประเมินอาการวิงเวียนศีรษะ (Dizziness Handicap Inventory: DHI) เป็นเครื่องมือที่ช่วยในการประเมินระดับความรุนแรงของอาการเวียนศีรษะที่รบกวนชีวิตประจำวันของผู้ป่วย แบบทดสอบประกอบด้วยคำถาม 25 ข้อ แบ่งออกเป็น 3 ด้าน ได้แก่ 1) คำถามประเมินด้านอาการ 7 ข้อ 2) คำถามประเมินด้านอารมณ์ 9 ข้อ และ 3) คำถามประเมินความสามารถในการใช้ชีวิตประจำวัน 9 ข้อ คำตอบแบ่งออกเป็น 3 ระดับ คือ มีอาการบ่อย ๆ (4 คะแนน) เป็นบางครั้ง (2 คะแนน) และไม่มีอาการ (0 คะแนน) ตามตาราง 1

ตาราง 1 แบบประเมินอาการวิงเวียนศีรษะ (Dizziness Handicap Inventory)

ข้อที่	อาการ อารมณ์ และความสามารถในการใช้ชีวิต	บ่อยๆ (4)	บางครั้ง(2)	ไม่มี (0)
P1	เมื่อฉันเงยหน้าขึ้นจะทำให้ฉันมีอาการเวียนหัวมากขึ้น			
E2	อาการเวียนหัวที่เป็นอยู่ ทำให้ฉันรู้สึกสับสนและเป็นกังวล			
F3	อาการเวียนหัวที่เป็นอยู่ ทำให้ฉันมีข้อจำกัดในการเดินทางไปทำธุระหรือพักผ่อนนอกบ้าน			
P4	เมื่อฉันเดินผ่านช่องทางแคบๆ เช่น ระหว่างชั้นวางของในร้านค้า มักทำให้เวียนหัวมากขึ้น			
F5	อาการเวียนหัวที่เป็นอยู่ ทำให้ฉันยากลำบากที่จะล้มตัวลงนอนหรือลุกขึ้นจากเตียงทันที			
F6	อาการเวียนหัวที่เป็นอยู่ ทำให้ฉันมีข้อจำกัดในการเข้าร่วมงานสังคมนอกบ้าน เช่น งานเลี้ยง งานสังสรรค์ ไปร่วมงานบ้านเพื่อน เป็นต้น			
F7	อาการเวียนหัวที่เป็นอยู่ ทำให้ฉันยากลำบากมากขึ้นในการอ่านหนังสือ เอกสารต่างๆ			
P8	การทำกิจกรรมที่ใช้แรง เช่น ออกกำลังกาย เต้น ทำงานบ้าน จะยิ่งทำให้ฉันเวียนหัวมากขึ้น			
E9	อาการเวียนหัวที่เป็นอยู่ ทำให้ฉันกลัวที่จะออกจากบ้านเพียงลำพังโดยไม่มีใครไปด้วย			
E10	จากอาการเวียนหัวที่ฉันเป็น ทำให้ฉันลำบากใจในการพบปะหรืออยู่ต่อหน้าผู้อื่น			
P11	การขยับศีรษะเร็วๆ ทำให้ฉันมีอาการเวียนหัวมากขึ้น			
F12	อาการเวียนหัวที่ฉันเป็นอยู่นี้ ทำให้ฉันต้องหลีกเลี่ยงการขึ้นที่สูง			
P13	เมื่อฉันนอนพลิกตัวบนเตียง ทำให้ฉันมีอาการเวียนหัวมากขึ้น			
F14	อาการเวียนหัวที่ฉันเป็น ทำให้ฉันยากที่จะทำงานบ้านหรือบริเวณรอบบ้านได้อย่างขันแข็ง			

ตาราง 1 (ต่อ)

ข้อที่	อาการ อารมณ์ และความสามารถในการใช้ชีวิต	บ่อยๆ (4)	บางครั้ง(2)	ไม่มี (0)
E15	จากที่ฉันมีอาการเวียนหัว ฉันจึงกลัวว่าคนอื่นอาจจะมองว่าฉันเป็นคนขี้เมาหรือติดยาได้			
F16	จากอาการที่ฉันเวียนหัว ทำให้ฉันยากลำบากที่จะเดินด้วยตนเองโดยไม่พึ่งอุปกรณ์ช่วย			
P17	เวลาที่ฉันเดินไปตามทางเท้า ฉันจะมีอาการเวียนหัวมากขึ้น			
E18	อาการเวียนหัวที่ฉันเป็น ทำให้ฉันรู้สึกยากลำบากที่จะมีสมาธิจดจ่อกับเรื่องใด ๆ ได้			
F19	อาการเวียนหัวนี้ ทำให้ฉันไม่สามารถเดินในท่ามกลางที่มืดๆ ได้แม้อยู่ในบริเวณบ้านก็ตาม			
E20	อาการเวียนหัวที่เป็นอยู่ ส่งผลให้ฉันกลัวที่จะอยู่บ้านเพียงลำพังคนเดียว			
E21	จากที่ฉันมีอาการเวียนหัว ทำให้ฉันรู้สึกเหมือนว่าฉันเป็นคนพิการ ช่วยตนเองไม่ค่อยได้			
E22	อาการเวียนหัว ทำให้ฉันรู้สึกเครียด เกรงว่าสัมพันธภาพในครอบครัวหรือกับเพื่อนจะแย่ลง			
E23	อาการเวียนหัวที่เป็นอยู่ ทำให้ฉันรู้สึกหดหู่ใจ ซึมเศร้า			
F24	อาการเวียนหัวที่ฉันเป็นอยู่นั้นรบกวนการทำงานบ้านหรืองานประจำของฉัน			
P25	การก้มหน้าโค้งลงทำให้ฉันมีอาการเวียนหัวมากขึ้น			

2.3 เทคนิคการเรียนรู้ของเครื่อง (Machine learning)

เทคนิคการเรียนรู้ของเครื่อง (Machine learning) คือ การเรียนรู้ของคอมพิวเตอร์โดยอาศัยข้อมูล เพื่อใช้ในการทำนาย หรือช่วยในการตัดสินใจได้ เทคนิคการเรียนรู้ของเครื่องสามารถแบ่งประเภทได้ 3 ประเภท คือ

1. การเรียนรู้แบบมีผู้สอน (Supervised learning)

เทคนิคการเรียนรู้ของเครื่องประเภทนี้ ข้อมูลที่ใช้ในการเรียนรู้ (training data) ของคอมพิวเตอร์จะระบุผลลัพธ์ (target value) ที่ต้องการไว้ โดยคอมพิวเตอร์จะเรียนรู้ความสัมพันธ์

ของข้อมูลที่ป้อนเข้า (input) กับผลลัพธ์ที่ออกมา (outcome) จากข้อมูลก่อนหน้าที่มี เพื่อให้สามารถทำนายหรือตัดสินใจได้

ปัญหาที่ใช้ Supervised machine learning สามารถแบ่งได้เป็น 2 แบบคือ

1.1 ปัญหาเชิงถดถอย (Regression) เป็นปัญหาที่ต้องการผลลัพธ์ออกมาเป็นค่าตัวเลข

1.2 ปัญหาเชิงแบ่งประเภท (Classification) เป็นปัญหาที่ต้องการผลลัพธ์ออกมาเป็นกลุ่มหรือประเภทของข้อมูล

2. การเรียนรู้แบบไม่มีผู้สอน (Unsupervised learning)

ตรงกันข้ามกับการเรียนรู้แบบมีผู้สอน คือ ข้อมูลที่ใช้ในการเรียนรู้ของคอมพิวเตอร์จะไม่มีการระบุผลลัพธ์ที่ต้องการไว้ โดยคอมพิวเตอร์จะเรียนรู้เพื่อจัดกลุ่มข้อมูลจากคุณสมบัติหรือลักษณะจำเพาะของข้อมูลที่คล้ายกันให้อยู่ด้วยกัน และสร้างขอบเขตเพื่อจำแนกข้อมูลเหล่านั้น

3. การเรียนรู้แบบเสริมกำลัง (Reinforcement learning)

จะคล้ายกับการเรียนรู้แบบไม่มีผู้สอน คือ จะใช้ข้อมูลที่ไม่มีการระบุผลลัพธ์ในการเรียนรู้ของคอมพิวเตอร์ แต่สิ่งที่ต่างกันคือ ในการเรียนรู้ของคอมพิวเตอร์จะมีคะแนนเข้ามาเกี่ยวข้องโดยจำคำนวณว่าในสถานการณ์นั้นจะต้องทำอย่างไรเพื่อให้ได้คะแนน หรือผลลัพธ์สูงที่สุด เช่น การคำนวณทิศทางการเดินหมากรุก เป็นต้น

2.4 อัลกอริธึมที่ใช้ในเทคนิคการเรียนรู้ของเครื่อง

อัลกอริธึมที่ใช้ในเทคนิคการเรียนรู้ของเครื่อง มีมากมาย ในที่นี้ขอกล่าวถึง Naïve Bayes, Support Vector Machine, K-Nearest Neighbors, Decision Tree และ Random Forest

2.4.1 Naïve Bayes

Naïve Bayes เป็นอัลกอริธึมสำหรับปัญหาเชิง Classification โดยใช้ความน่าจะเป็นที่เรียกว่า conditional probability คือ

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \quad (1)$$

เมื่อ $P(A|B)$ คือ conditional probability หรือ ความน่าจะเป็นของเหตุการณ์ A เมื่อเกิดเหตุการณ์ B แล้ว

$P(A \cap B)$ คือ ค่า joint probability หรือค่าความน่าจะเป็นที่เหตุการณ์ A และเหตุการณ์ B เกิดขึ้นร่วมกัน

$P(B)$ คือ ความน่าจะเป็นที่จะเกิดเหตุการณ์ B

ซึ่งขณะเดียวกันจะได้

$$P(B|A) = \frac{P(A \cap B)}{P(A)} \quad (2)$$

เมื่อนำทั้งสองสมการมารวมกันจะได้สมการที่เรียกว่า Bayes theorem ซึ่งนำไปใช้ในการสร้างแบบจำลองเพื่อทำนายข้อมูลต่อได้

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} \quad (3)$$

ดังนั้น การใช้ Naïve Bayesในปัญหาเชิง Classification ในการจำแนกประเภทข้อมูล ตาม Bayes theorem จะได้ดังนี้

ถ้า $P(A_1|B) > P(A_2|B)$ แล้ว ข้อมูลจะถูกจัดเป็นประเภท A_1

ถ้า $P(A_1|B) < P(A_2|B)$ แล้ว ข้อมูลจะถูกจัดเป็นประเภท A_2

ข้อดีของการใช้ Naïve Bayes ในการแก้ปัญหาเชิง Classification คือ เป็นอัลกอริธึมที่เหมาะกับข้อมูลขนาดเล็ก และสามารถจัดการกับข้อมูลที่มีหลายประเภทในการจำแนกได้ แต่การใช้อัลกอริธึมนี้จำเป็นต้องจัดเตรียมข้อมูล เนื่องจากจะใช้ได้กับข้อมูลที่เป็นตัวเลข และบูลีนเท่านั้น

2.4.2 Support Vector Machine (SVM)

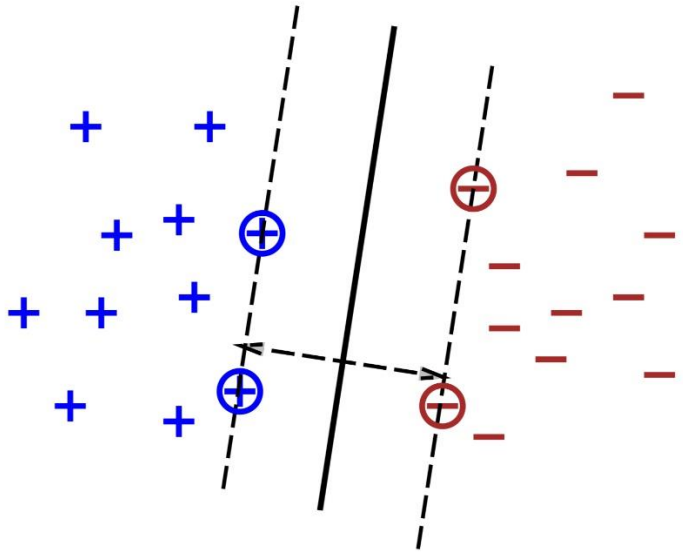
Support Vector Machine (SVM) เป็นอัลกอริธึมที่ใช้ในงาน Machine Learning สำหรับปัญหาทั้ง Classification และ Regression โดยมีหลักการคือ การสร้างเส้น (Hyperplane) สำหรับแบ่งข้อมูลที่กระจายอยู่บน feature space ออกจากกัน และวัดความคล้ายของข้อมูลด้วยเคอร์เนลฟังก์ชัน (Kernel Function) โดยจะต้องหาเส้นตรงที่ดีที่สุดในการแบ่งข้อมูล โดยดูจากระยะ (margin) จากเส้นแบ่งไปยังข้อมูลที่ใกล้ที่สุด เส้นแบ่งใดที่มีระยะมากที่สุด (Maximum margin) จะเป็นเส้นแบ่งที่สามารถแบ่งข้อมูลได้ดีที่สุด ดังภาพประกอบ 2 เส้นสำหรับแบ่งข้อมูลสามารถคำนวณได้จากสมการ (4)

$$f(x) = wx + b \quad (4)$$

เคอร์เนลฟังก์ชันที่นิยมใช้มี 3 ชนิด ได้แก่

- Polynomial
- Radial Basis Function (RBF)
- Sigmoid

ข้อดีของการใช้อัลกอริธึม SVM ในการแก้ปัญหาเชิง Classification คือ มีข้อผิดพลาดน้อย ไม่เปลืองทรัพยากรคอมพิวเตอร์ และง่ายต่อการแปลผล แต่ต้องระมัดระวังการปรับพารามิเตอร์ และเลือกชนิดเคอร์เนลฟังก์ชันให้เหมาะสมกับข้อมูลที่นำมาใช้



ภาพประกอบ 2 แสดงการแบ่งกลุ่มข้อมูลประเภทบวกและลบ ด้วยเส้นแบ่งที่มีระยะมากที่สุด (maximum margin)

ที่มา : Bottou L, Lin CJ. Support vector machine solvers. Large scale kernel machines. 2007;3(1):301-20.

2.4.3 K-Nearest Neighbors (KNN)

K-Nearest Neighbors เป็นอัลกอริธึมที่ใช้ในการจำแนกประเภทของข้อมูล โดยเปรียบเทียบข้อมูลใหม่กับข้อมูลเดิม (neighbors) ว่าใกล้เคียงกับข้อมูลจำนวน K ตัว ประเภทใดมากที่สุด ดังภาพประกอบ 3 ดังนั้น พารามิเตอร์หลักที่ใช้ในการปรับค่า คือ

- n_neighbors คือ จำนวนข้อมูลใกล้เคียงที่ใช้ในการเปรียบเทียบกับข้อมูลใหม่
- weights คือ ระยะห่างของข้อมูลใกล้เคียงกับข้อมูลใหม่ หากใช้ ‘uniform’ หมายถึง จะไม่นำระยะห่างของข้อมูลมาคิดคำนวณหาประเภทของข้อมูลด้วย แต่หากใช้ ‘distance’ หมายถึง จะนำระยะห่างของข้อมูลมาใช้ โดยข้อมูลเดิมที่อยู่ใกล้กว่า จะมีผลต่อการจัดประเภทให้กับข้อมูลใหม่มากกว่า ข้อมูลเดิมที่อยู่ห่างออกไป
- p คือ การคำนวณระยะห่างของข้อมูล มี 2 แบบคือ

ถ้า p = 1 จะคำนวณระยะห่างแบบ Manhattan distance ดังสมการ (5)

$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i| \qquad (5)$$

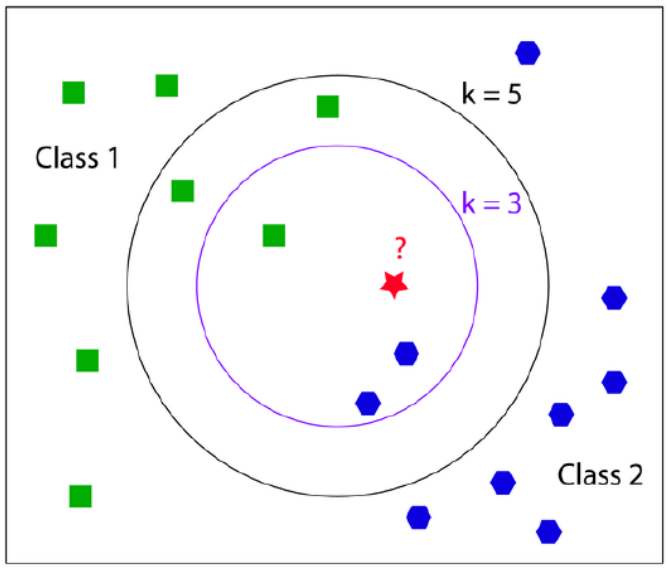
ถ้า p = 2 จะคำนวณระยะห่างแบบ Euclidean distance ดังสมการ (6)

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (6)$$

เมื่อ $d(x,y)$ คือ ระยะห่างของข้อมูล

n คือ มิติของข้อมูล

ข้อดีของการใช้อัลกอริธึม KNN ในการแก้ปัญหาเชิง Classification คือไม่ไวต่อค่าที่ผิดปกติของข้อมูล และให้ค่า accuracy สูงแต่เนื่องจากในการจำแนกประเภทข้อมูลแต่ละครั้งต้องคำนวณหาระยะห่างระหว่างข้อมูลในทุก ๆ จุด จึงทำให้สิ้นเปลืองทรัพยากรคอมพิวเตอร์



ภาพประกอบ 3 แสดงการจำแนกประเภทข้อมูลโดยอัลกอริธึม KNN เมื่อกำหนดค่า k ต่าง ๆ

ที่มา : Cover TM, Hart PE. Nearest neighbor pattern classification. IEEE transactions on information theory. 1967 Jan 27;13(1):21-7.

2.4.4 Decision Tree

Decision Tree เป็นอัลกอริธึมที่มีหลักการจากการสร้างกฎการตัดสินใจเพื่อแบ่งกลุ่มข้อมูลในรูปแบบเชิงลำดับชั้น (hierarchical classifier) [6] กลุ่มของข้อมูลที่ได้จากการแบ่งจะถูกแบ่งด้วยกฎอีกไปเรื่อย ๆ จนกระทั่งได้ประเภทของข้อมูลที่ต้องการจำแนก ซึ่งการแบ่งกลุ่มในลักษณะนี้จะมีลักษณะคล้ายแผนภาพต้นไม้

ข้อดีของ Decision Tree คือ ไม่เปลืองทรัพยากรคอมพิวเตอร์ สามารถใช้กับข้อมูลที่มีค่าไม่ครบถ้วนได้ และเป็นอัลกอริธึมที่เข้าใจง่าย แต่หากปรับพารามิเตอร์ไม่เหมาะสมก็อาจทำให้เกิดปัญหา overfitting ได้ง่ายเช่นกัน

2.4.5 Random Forest

Random Forest เป็นการสุ่มข้อมูลแบบ bootstrap (สุ่มข้อมูลแบบเลือกซ้ำได้) เพื่อที่จะสร้างต้นไม้แต่ละต้น แล้วให้ต้นไม้มีการเรียนรู้และแตกกิ่งจนสามารถทำนายถูกหมด ซึ่งในการทำนาย ต้นไม้แต่ละต้นจะมีการสุ่มชุดของ feature มาใช้ในการทำนาย ดังนั้นจึงทำให้ได้ต้นไม้ที่ไม่เหมือนกัน โดยในการทำนายประเภทของข้อมูลจะใช้ต้นไม้หลาย ๆ ต้นมารวมกันเพื่อหาความน่าจะเป็นของประเภทของข้อมูลที่ต้องการทำนาย

พารามิเตอร์หลักที่สามารถปรับค่าได้มีดังนี้

- max_features: จำนวน features ที่จะสุ่มเลือกเพื่อใช้ในการสร้างต้นไม้และแตกกิ่งออกไป
- max_depth: สามารถกำหนดความซับซ้อนของแบบจำลอง decision tree
- min_samples_split: จำนวน samples อย่างน้อยใน node ที่จะให้ต้นไม้สามารถแตกกิ่งต่อไปได้
- bootstrap: เลือกเปิด mode ได้ว่าจะใช้การสุ่มหรือไม่ โดย default จะเป็น True คือเปิดการสุ่ม
- oob_score: เมื่อเปิด mode การสุ่มแบบ bootstrap จะสามารถใช้ out-of-bag samples ในการคำนวณเพื่อหาความถูกต้อง (accuracy) ได้ ในการเปิด mode นี้จะทำให้มีการคำนวณเพื่อหา out-of-bag error ที่น้อยที่สุดมาใช้

ข้อดีของการแก้ปัญหาเชิง Classification ด้วยอัลกอริธึม Random forest คือ เกิดปัญหา Overfitting ได้น้อยกว่าอัลกอริธึม Decision Tree และมีประสิทธิภาพสูง แต่มีข้อเสียคืออัลกอริธึมมีความซับซ้อน และหากใช้จำนวน Decision Tree ในอัลกอริธึม Random Forest มากจะทำให้ใช้เวลาในการประมวลผลมาก

2.5 การแบ่งข้อมูลสำหรับทดสอบแบบจำลอง

ในเทคนิคการเรียนรู้ของเครื่องนั้น เครื่องจะเรียนรู้จากข้อมูลโดยข้อมูลจะถูกแบ่งเป็นข้อมูลสำหรับสร้างแบบจำลอง (training set) และข้อมูลสำหรับทดสอบแบบจำลอง (test set) เทคนิคการแบ่งข้อมูลมี 2 ประเภท ดังนี้

2.5.1 Train/Test Split คือ การแบ่งข้อมูล 2 ส่วนสำหรับสร้างแบบจำลอง และอีกส่วนสำหรับทดสอบแบบจำลอง เช่น แบ่ง 80% สำหรับสร้างแบบจำลอง และแบ่ง 20% สำหรับทดสอบแบบจำลอง เป็นต้น

2.5.2 Cross-Validation [7] เป็นหลักการการแบ่งข้อมูลสำหรับใช้ในการสร้างและทดสอบประสิทธิภาพแบบจำลองที่ได้จากการเรียนรู้ของเครื่องที่ได้รับความนิยม หลักการของ Cross-Validation คือ การแบ่งข้อมูล n ส่วน โดยใช้ n-1 ส่วนสำหรับเป็นข้อมูลในการเรียนรู้ของเครื่อง (the training sample) และ 1 ส่วนสำหรับเป็นข้อมูลในการทดสอบแบบจำลอง (the validation sample) เช่น 5-fold cross-validation คือ การแบ่งข้อมูลทั้งหมด 5 ส่วนเท่า ๆ กัน ข้อมูล 4 ส่วนจะใช้ในการสร้างแบบจำลอง และจะวนใช้ข้อมูลจนครบ คือ ทั้งหมด 5 ครั้ง ดังนั้นการทดสอบประสิทธิภาพจึงครอบคลุมข้อมูลทั้งหมด ทำให้ผลการทดสอบประสิทธิภาพมีความน่าเชื่อถือ

2.6 การคัดเลือกฟีเจอร์ (Feature selection)

ในงาน Machine Learning การคัดเลือกฟีเจอร์มีความสำคัญในการเพิ่มประสิทธิภาพในการสร้างแบบจำลอง ลดเวลาในการเรียนรู้ของคอมพิวเตอร์ และช่วยลดปัญหาการสร้างโมเดลที่ซับซ้อนเกินไป (overfitting) ได้

เทคนิคในการเลือกฟีเจอร์ [8, 9] แบ่งออกเป็น 3 ประเภท คือ

2.6.1 Filter techniques คือ การเลือกฟีเจอร์โดยดูเฉพาะคุณสมบัติภายในของข้อมูลว่ามีความเกี่ยวข้องระหว่างฟีเจอร์นั้นและผลลัพธ์ (outcome) มากเท่าไร โดยฟีเจอร์ที่มีความเกี่ยวข้องน้อยจะถูกคัดออกไม่นำมาใช้สร้างแบบจำลอง ข้อดีของ Filter techniques คือ เป็นเทคนิคที่ใช้ได้รวดเร็ว เหมาะกับข้อมูลที่มีฟีเจอร์ปริมาณมาก เพราะไม่ต้องใช้อัลกอริธึมในการสร้างแบบจำลองมาเกี่ยวข้อง และทำการคำนวณหาความเกี่ยวข้องของฟีเจอร์เพียงครั้งเดียว จึงทำให้ไม่ต้องใช้ทรัพยากรในการคำนวณ แต่ก็มีข้อเสียคือ ฟีเจอร์แต่ละตัวจะถูกวัดคะแนนความเกี่ยวข้องแยกกันกับการสร้างแบบจำลอง ดังนั้นฟีเจอร์ที่ได้อาจไม่เหมาะที่จะนำไปใช้ในการสร้างแบบจำลองก็ได้ ตัวอย่างของ Filter techniques เช่น การคำนวณ correlation, Chi-square, t-test, information gain เป็นต้น

2.6.2 Wrapper method เป็นการคัดเลือกฟีเจอร์โดยคำนวณจากการเรียนรู้และทดสอบของคอมพิวเตอร์ในการสร้างแบบจำลองด้วยอัลกอริธึมที่เลือกไว้ ดังนั้นจึงมีข้อดีคือเป็นการหาฟีเจอร์ที่เหมาะสมในการสร้างแบบจำลองโดยตรงเลย แต่ก็มีข้อเสียคือ มีความเสี่ยงที่จะเกิดโมเดลที่ซับซ้อนเกินไป (overfiting) มากกว่า Filter techniques มีการคำนวณหลายครั้ง และใช้เวลาในการคำนวณมากกว่า Filter techniques ตัวอย่างของ Wrapper method คือ

2.6.2.1 Forward selection หรือการเพิ่มฟีเจอร์ทีละตัว เพื่อดูว่าแบบจำลองมีประสิทธิภาพดีขึ้นหรือไม่ หากมีประสิทธิภาพดีขึ้นก็จะเก็บฟีเจอร์นั้นไว้

2.6.2.2 Recursive Feature Elimination เป็นวิธีการเลือกฟีเจอร์ โดยจะเอาฟีเจอร์ทั้งหมดเข้าไปใช้ ในการสร้างแบบจำลอง ในแต่ละรอบของการสร้างแบบจำลองจะคัดฟีเจอร์ที่มีผลต่อการสร้าง แบบจำลองน้อยที่สุดออก แล้วใช้ฟีเจอร์ที่เหลือสร้างแบบจำลองใหม่ ทำวนรอบจนเหลือจำนวนฟีเจอร์ที่ต้องการ

2.6.3 Embedded method เป็นการหาฟีเจอร์ที่เหมาะสมที่สามารถสร้างแบบจำลองให้ได้มีประสิทธิภาพมากที่สุดขณะที่กำลังทำแบบจำลอง ตัวอย่างเช่น decision tree, L1 regularization ข้อดีคือ ได้ฟีเจอร์ที่มีความเกี่ยวข้องกับการสร้างแบบจำลองสูง และใช้เวลาในการคำนวณน้อยกว่า Wrapper method

2.7 การสกัดฟีเจอร์ (Feature Extraction)

เทคนิคการสกัดข้อมูล (Feature Extraction) ใช้ในการจัดการข้อมูลที่มีจำนวนตัวแปรหรือฟีเจอร์มาก ด้วยการแปลงตัวแปรเดิมให้เป็นตัวแปรใหม่ที่มีมิติลดลง ทำให้ลดการใช้ทรัพยากรทางคอมพิวเตอร์ และเพิ่มประสิทธิภาพในการจำแนกข้อมูล แต่ก็มีข้อเสียคือข้อมูลฟีเจอร์บางส่วนจะมีการสูญหายไปหลังจากการสกัดฟีเจอร์

งานวิจัยนี้ได้ใช้เทคนิคการวิเคราะห์องค์ประกอบหลัก (Principal Component Analysis: PCA) [10] ในการสกัดฟีเจอร์ โดยหลักการของ PCA คือการลดตัวแปรจากการสร้างตัวแปรใหม่ขึ้นมาซึ่งจะสกัดหรือดึงรายละเอียดข้อมูลสำคัญจากตัวแปรเดิมมาไว้ให้ได้มากที่สุด ตัวอย่างเช่นให้ X เป็นเมตริกซ์ของฟีเจอร์ โดยมีขนาด m แถว หรือจำนวนตัวอย่าง และมี n คอลัมน์ หรือ จำนวนฟีเจอร์ สามารถลดขนาดของเมตริกซ์ X ได้โดยขั้นตอนดังนี้

1. นอมัลไลซ์ข้อมูล โดยใช้ค่าเบี่ยงเบนมาตรฐานดังสมการ (7) จะได้

$$X_{norm} = \frac{[X - mean(X)]}{std(x)} \quad (7)$$

2. นำข้อมูลเมตริกซ์ที่ผ่านการนอมัลไลซ์มาหาค่าสหสัมพันธ์ ได้เป็นเมตริกซ์ R
3. นำเมตริกซ์ R ไปสกัดคุณลักษณะเด่น โดยการหาค่าไอเกน (Eigenvalue) โดยจะสนใจค่าไอเกนที่ไม่ใช่ 0 เท่านั้น และเรียงลำดับค่าไอเกนจากมากไปน้อย
4. เลือกไอเกนเวกเตอร์ที่ให้ค่าไอเกนสูงตามลำดับตามที่ต้องการลดขนาดตัวแปร เช่น ถ้าเลือกมา k ตัว ดังนั้นข้อมูลที่เหลือจากการทำ PCA จะเป็นเมตริกซ์ที่มีขนาด m แถว k คอลัมน์

2.8 งานวิจัยที่เกี่ยวข้องกับความสัมพันธ์ของแบบประเมินอาการวิงเวียนศีรษะและโรคเวียนศีรษะขณะเปลี่ยนท่า

ในปัจจุบันมีงานวิจัยที่แสดงความสามารถของแบบประเมินอาการวิงเวียนศีรษะ (Dizziness Handicap Inventory: DHI) ในการจำแนกประเภทของผู้ป่วยที่มีอาการวิงเวียนศีรษะจากโรคเวียนศีรษะขณะเปลี่ยนท่า และผู้ป่วยที่มีอาการวิงเวียนศีรษะจากโรคอื่น ตัวอย่างเช่น

2.8.1 บทความวิจัยเรื่อง Epidemiology of benign paroxysmal positional vertigo: a population based study โดย M von Brevern, A Radtke และคณะ

งานวิจัยนี้เป็นการศึกษาหาความชุก อุบัติการณ์การเกิดโรค และความรุนแรงของอาการเวียนศีรษะในระดับต่างๆ ของโรค BPPV ในประเทศเยอรมนี จากการสอบถามและสำรวจ [11] โดยพบว่า ในผู้ป่วยที่มารักษาด้วยอาการเวียนศีรษะ มีสาเหตุส่วนใหญ่เกิดจากโรค BPPV หรือประมาณ 20 % โดยผู้ป่วย BPPV เมื่อได้รับการรักษาแล้วก็ยังมีโอกาสที่จะกลับมาเป็นใหม่ได้

BPPV มีโอกาสเกิดในผู้ใหญ่และผู้สูงอายุ ได้มากกว่าวัยรุ่นหรือเด็ก โดยอายุเฉลี่ยของผู้ป่วย BPPV คือ 49 ปี และพบบ่อยในผู้ป่วยอายุ 80 ปีขึ้นไป โดยอาการของโรคมีตั้งแต่ เวียนศีรษะ มองไม่ชัด คลื่นไส้ ทรงตัวไม่ได้ หรือตื่นขึ้นมากลางดึกเนื่องจากโรค BPPV

2.8.2 บทความวิจัยเรื่อง Performance of DHI Score as a Predictor of Benign Paroxysmal Positioning Vertigo in Geriatric Patients with Dizziness/Vertigo: A Cross-Sectional Study โดย Amrish Saxena และ Manish Chandra Prabhakar

งานวิจัยนี้ประเมินผลกระทบของอาการเวียนศีรษะต่อชีวิตประจำวันของผู้ป่วย และประเมินความสามารถในการทำนายโรค BPPV จากคะแนนจากแบบทดสอบ DHI [12] โดยใช้ข้อมูลคะแนนรวมจากแบบทดสอบ DHI ของผู้ป่วยสูงอายุชาวอินเดียที่เข้ารับการรักษาในคลินิกผู้ป่วยนอกในโรงพยาบาลวิทยาลัยแพทย์ของประเทศอินเดียด้วยอาการเวียนศีรษะ เมื่อผู้ป่วยมารับการรักษาจะทำการประเมินด้วยแบบทดสอบ DHI ก่อนตรวจร่างกาย และซักประวัติผู้ป่วย ในขั้นตอนที่สอง ผู้ป่วยจะได้รับการตรวจ Dix-Hallpike ซึ่งเป็นวิธีมาตรฐานสำหรับวินิจฉัยโรค BPPV โดยแพทย์ผู้เชี่ยวชาญ ซึ่งไม่ทราบคะแนนจากแบบทดสอบ DHI ที่ผู้ป่วยได้ทำไปก่อนหน้า และในขั้นตอนสุดท้ายผู้ป่วยจะได้รับการตรวจสมรรถภาพทางการได้ยิน เพื่อประเมินความผิดปกติอื่นของหู และได้รับการวินิจฉัยโรคโดยแพทย์ผู้เชี่ยวชาญในท้ายที่สุด

ผู้ป่วยที่ได้รับการประเมินด้วยแบบทดสอบ DHI แบ่งเป็น 2 กลุ่มคือ ผู้ป่วยที่มารักษาอาการเวียนศีรษะเนื่องจากโรค (BPPV Group) และ ผู้ป่วยที่มารับการรักษาอาการเวียนศีรษะเนื่องจากโรคอื่นที่ไม่ใช่โรค BPPV (Non-BPPV Group) ผู้วิจัยได้ประเมินค่าเฉลี่ยคะแนนรวมจากแบบทดสอบ DHI ของผู้ป่วยทั้งสองกลุ่มเพื่อนำมาเปรียบเทียบกัน และหาค่าที่ดีที่สุด (cut-off) ในการทำนายโรค BPPV โดยพบว่าค่าเฉลี่ยคะแนนรวมของแบบทดสอบ DHI ในผู้ป่วยกลุ่มที่เป็นโรค

BPPV สูงกว่าคะแนนรวมของแบบทดสอบ DHI ในผู้ป่วยกลุ่มที่ไม่ได้เป็นโรค BPPV และ cut-off คะแนนรวมจากแบบทดสอบ DHI เท่ากับ 50 คะแนน เป็นค่า cut-off ที่ดีที่สุดในการทำนายโรค BPPV โดยให้ค่า Area Under Curve (AUC) 0.982, Sensitivity 94.7% และ Specificity 94.2% ซึ่งจากผลการวิจัยครั้งนี้ผู้วิจัยจึงสรุปว่า แบบทดสอบ DHI เป็นเครื่องมือที่มีประโยชน์ที่สามารถช่วยในการวินิจฉัยโรค BPPV ได้

2.8.3 บทความวิจัยเรื่อง Validation of 5-item & 2-item questionnaires in Chinese version of Dizziness Handicap Inventory for screening objective benign paroxysmal positional vertigo. โดย Wei Chen และคณะ

งานวิจัยนี้เปรียบเทียบผลคะแนนของ DHI ในผู้ป่วยชาวจีน 55 คน ที่มาด้วยอาการเวียนศีรษะจากโรค BPPV และ 58 คน ที่มีอาการเวียนศีรษะจากโรคอื่นที่ไม่ใช่ BPPV โดยใช้ผลคะแนนรวม DHI ทั้งแบบสอบถาม [13] ผลคะแนนรวมจาก 5 คำถามย่อยด้าน Physical ในแบบทดสอบ DHI และ ผลคะแนนรวมจาก 2 คำถามย่อยด้าน Physical ในแบบทดสอบในการวิเคราะห์ โดยผู้ป่วยจะถูกวินิจฉัยโรค BPPV ด้วยการตรวจ Dix-Hallpike

5 คำถามย่อย และ 2 คำถามย่อยที่ถูกแยกออกมาเพื่อใช้ในการคิดคะแนน เป็นคำถามในหมวด Physical ของแบบทดสอบ DHI ซึ่งเป็นการดูการเกิดอาการเวียนศีรษะเมื่อทำท่าทางต่างๆ โดยใน 5 คำถามย่อย ได้แก่ เงยหน้า ก้ม หันศีรษะอย่างรวดเร็ว นอนราบ และกลิ้งตัว 2 คำถามย่อย ได้แก่ นอนราบ และกลิ้งตัว ผลจากการวิจัยพบว่าคะแนนจากแบบทดสอบ DHI ทั้งแบบรวมคะแนนทั้งแบบทดสอบ และรวมเฉพาะคำถามย่อยตามที่ได้แยกมา ต่างก็มีประสิทธิภาพในการช่วยทำนาย BPPV ทั้งสิ้น โดยการใช้คะแนนรวมจากคำถามย่อยทั้งแบบ 5 คำถาม และ 2 คำถาม ในแบบทดสอบ DHI ในการทำนาย BPPV ให้ประสิทธิภาพมากกว่าการใช้คะแนนรวมจากแบบทดสอบ DHI ทั้งหมด และเมื่อประเมินความสอดคล้องภายในของแบบทดสอบด้วย cronbach α พบว่าแบบทดสอบย่อย 5 คำถามจาก DHI ให้ค่าความสอดคล้องภายในดีที่สุด

จากการศึกษางานวิจัยที่เกี่ยวข้อง จะเห็นได้ว่าแบบประเมินอาการวิงเวียนศีรษะมีประสิทธิภาพในการทำนายจำแนกผู้ป่วยที่มีอาการวิงเวียนศีรษะเนื่องจากโรคเวียนศีรษะขณะเปลี่ยนท่า จากผู้ป่วยที่มีอาการวิงเวียนศีรษะจากโรคอื่น ๆ แต่ยังไม่พบงานวิจัยที่ศึกษาการใช้แบบประเมินอาการวิงเวียนศีรษะจำแนกประเภทของโรคเวียนศีรษะขณะเปลี่ยนท่า ดังนั้นหากสามารถสร้างแบบจำลองการทำนายจำแนกประเภทโรคเวียนศีรษะขณะเปลี่ยนท่าจากแบบประเมินอาการวิงเวียนศีรษะได้อย่างมีประสิทธิภาพ จะทำให้มีประโยชน์ทางการแพทย์อย่างยิ่ง

บทที่ 3
วิธีดำเนินการวิจัย

ในการวิจัยครั้งนี้ ผู้วิจัยได้ดำเนินการตามขั้นตอนดังนี้

1. การกำหนดประชากรและการสุ่มกลุ่มตัวอย่าง
2. การสร้างเครื่องมือที่ใช้ในการวิจัย
3. การเก็บรวบรวมข้อมูล
4. การจัดกระทำและการวิเคราะห์ข้อมูล

3.1 การกำหนดประชากรและการสุ่มกลุ่มตัวอย่าง

3.1.1 ประชากร

ข้อมูลคะแนน DHI จากผู้ป่วยไทย 114 คนในปี พ.ศ. 2560 ที่มีการเก็บคะแนน DHI ก่อนการรักษา และได้รับการวินิจฉัยโรคว่าเป็น BPPV ชนิด PC-BPPV หรือชนิด HC-BPPV โดยแพทย์เฉพาะทาง จากนั้นมีการรักษาและติดตามจนหายจากโรคแล้ว

3.1.2 การเลือกกลุ่มตัวอย่าง

ข้อมูลจะถูกแบ่งเป็น 2 ชุด คือ ชุดข้อมูลสำหรับสร้างแบบจำลอง และชุดข้อมูลสำหรับทดสอบแบบจำลอง ในอัตราส่วน 80:20

3.2 การสร้างเครื่องมือที่ใช้ในการวิจัย

เครื่องมือที่ใช้ในการวิจัยคือแบบประเมินอาการวิงเวียนศีรษะ โดย นพ.วิศาล มหาสิทธิวัฒน์ ซึ่งปรับมาจาก Dizziness Handicap Inventory (DHI) [14]

3.3 การเก็บรวบรวมข้อมูล

ในงานวิจัยนี้ได้ใช้ข้อมูลคะแนน DHI จากผู้ป่วยไทยในปี พ.ศ. 2560 ที่มีการเก็บคะแนน DHI ก่อนการรักษา และได้รับการวินิจฉัยโรคว่าเป็น BPPV ชนิด PC-BPPV หรือชนิด HC-BPPV โดยแพทย์เฉพาะทาง จากนั้นมีการรักษาและติดตามจนหายจากโรคแล้ว

ข้อมูลที่เก็บมีฟีเจอร์ 3 กลุ่มคือ

3.3.1 กลุ่มข้อมูลผู้ป่วย มีฟีเจอร์ดังนี้

- ระยะเวลาการรักษา
- อายุ

3.3.2 กลุ่มข้อมูลจาก DHI ตามตาราง 1

3.3.3 ฟีเจอร์ที่สร้างเพิ่มจากแบบทดสอบ

โดยในข้อมูลจะระบุการวินิจฉัยว่าเป็น BPPV ชนิดอะไรไว้ เพื่อใช้เป็น label สำหรับการทำ Machine learning ด้วย

3.4 การจัดกระทำข้อมูล และการวิเคราะห์ข้อมูล

งานวิจัยนี้ใช้ภาษาไพทอน (Python) ในการวิเคราะห์ข้อมูลและดำเนินการในงาน Machine Learning โดยมีขั้นตอนดังนี้

3.4.1 จัดการข้อมูลให้อยู่ในรูปแบบไฟล์ที่เหมาะสมคือ เก็บข้อมูลในรูปแบบไฟล์เวิร์กชีต Microsoft Excel (.xlsx) เพื่อให้สามารถนำมาใช้ในงาน Machine Learning ตัวอย่างของข้อมูลที่ใช้ดังภาพประกอบ 4

no	onset	nyst	age	P1	E2	F3	P4	F5	F6	F7	P8	E9	E10	P11	F12	P13	F14	E15	F16	P17	E18	F19	E20	E21	E22	E23	F24	P25	diagnosis
1	7	80	74	0	2	0	0	4	2	4	2	4	0	0	4	0	2	0	4	2	0	2	2	2	0	4	2	2	hcb
2	10	6.5	40	4	0	0	2	2	0	2	4	0	0	4	4	2	2	0	0	0	0	0	0	0	0	0	0	2	hcb
3	30	24.2	60	2	4	2	0	2	2	4	2	2	0	2	4	4	4	0	0	2	2	0	2	0	0	0	2	2	hcb
4	40	5	87	4	4	4	4	4	4	4	4	4	4	4	4	4	4	2	2	2	4	4	4	2	2	4	4	4	hcb
5	21	blind	37	2	4	0	0	4	0	0	2	2	0	4	2	4	0	0	0	2	2	0	0	0	0	2	4	4	hcb
6	30	87.6	54	4	4	4	4	4	4	4	4	4	4	4	4	4	4	2	4	4	4	2	2	2	4	4	4	4	hcb
7	4		52	4	0	0	0	2	0	0	0	0	0	4	0	2	4	0	0	0	0	0	0	0	0	2	0	4	hcb
8	15		55	4	0	4	2	4	4	2	2	2	2	2	2	4	2	0	2	2	2	2	2	2	4	2	4	4	hcb
9	4	2.2	47	2	2	2	2	4	0	4	4	2	0	4	4	2	2	0	0	2	2	2	2	0	0	0	2	4	hcb
10	3		79	2	2	2	2	4	2	2	4	4	2	4	0	4	2	0	4	2	2	2	4	2	2	2	0	2	hcb
11	7		82	4	2	4	4	4	4	4	4	4	4	4	4	4	4	4	4	0	2	4	4	4	2	2	4	4	hcb
12	14		43	0	4	2	2	2	2	4	0	0	0	2	4	2	0	0	0	2	2	2	0	0	0	2	2	2	hcb
13	3		59	2	0	2	0	2	0	2	0	2	0	2	2	2	4	0	2	0	0	0	0	0	0	0	2	2	hcb
14	0.3	20	54	4	4	4	0	4	4	4	2	4	0	4	0	4	4	0	0	0	4	0	4	0	0	2	4	4	hcb
15	30		61	2	4	2	2	2	0	2	2	0	0	2	4	2	0	0	0	2	0	0	0	0	0	0	0	4	hcb
16	8	1.4	55	4	0	2	2	4	2	2	2	0	0	0	0	2	2	0	4	0	2	0	0	0	0	0	2	2	hcb
17	35	17.3	63	4	2	2	2	4	2	2	4	2	0	4	2	4	2	0	0	2	2	2	2	0	0	0	2	4	hcb
18	2	60.7	35	2	2	4	0	2	2	2	0	2	0	4	0	4	2	0	0	0	2	0	2	0	0	2	2	2	hcb
19	2		33	2	2	2	0	2	2	2	2	2	0	4	0	2	2	0	0	0	2	0	2	0	0	2	2	2	hcb
20	7		80	0	0	2	0	2	2	2	2	2	2	4	2	0	2	0	4	2	4	4	2	0	2	2	4	2	hcb
21	14		40	4	4	4	0	4	2	4	2	2	2	4	4	4	2	0	2	2	4	2	4	0	0	2	2	4	hcb
22	4		55	2	4	0	4	4	4	2	4	4	0	0	0	0	2	4	4	2	2	0	0	0	4	2	0	0	hcb
23	7		31	4	4	4	4	2	4	4	4	0	0	4	0	4	4	0	2	0	4	0	4	4	2	4	4	2	hcb
24	10		66	4	2	2	2	4	2	2	4	2	2	4	4	4	4	0	4	2	2	4	4	2	2	2	2	4	hcb
25	5		54	2	0	4	2	4	4	4	4	4	4	4	4	4	4	0	2	2	2	4	4	2	2	2	4	4	hcb
26	5		80	2	0	4	0	2	2	4	0	4	2	2	4	2	4	2	4	2	4	4	4	4	4	2	4	4	hcb
27	1		57	2	2	2	0	4	2	2	2	2	2	4	2	4	2	2	2	2	2	0	2	0	0	0	2	2	hcb
28	14		65	2	4	2	0	4	0	2	4	2	0	2	2	4	2	0	0	0	2	0	0	0	0	2	4	4	hcb
29	10		66	0	2	4	0	4	0	0	0	2	0	2	0	4	0	0	0	0	0	0	4	0	2	0	0	0	hcb

ภาพประกอบ 4 ตัวอย่างข้อมูลที่ใช้สำหรับสร้างแบบจำลอง

3.4.2 ศึกษาฟีเจอร์ที่อยู่ในข้อมูล จัดระเบียบข้อมูลให้ง่ายต่อการทำแบบจำลองโดยจัดรูปแบบให้ข้อมูลเป็น DataFrame ดังภาพประกอบ 5 กำจัด outliers ที่อยู่ในข้อมูล โดยการหาค่า null ในข้อมูล และตัดคอลัมน์ที่มีค่า null สูงออก ดังภาพประกอบ 6 แปลงข้อมูลที่อยู่ในรูปข้อความให้เป็นตัวเลข จากนั้นแบ่งข้อมูลออกเป็นข้อมูลที่ใช้ฝึก และผลลัพธ์ ดังภาพประกอบ 7

```
os.chdir('C:\\Users\\User\\Desktop')
file_name = 'bppv dataset.xlsx'
df = pd.read_excel(file_name)
df
```

ภาพประกอบ 5 ตัวอย่างโค้ดนำเข้าไฟล์ข้อมูล และปรับรูปแบบเป็น DataFrame

```
(df.isnull().sum()).sort_values(ascending=False)
```

```
df.drop("nyst",axis=1,inplace=True)
```

ภาพประกอบ 6 ตัวอย่างโค้ดการหาค่า null ในข้อมูล และการตัดคอลัมน์ที่มีจะนวน null มากเกินไป

```
df['diagnosis'].replace(['hcb','pcb'],[1,0],inplace=True) #แปลงข้อมูลชนิดของโรคเรียนศีรษะรอบและเปลี่ยนค่าจากข้อความให้เป็นตัวเลข
```

```
df = df[['no', 'onset', 'age', 'P1', 'E2', 'F3', 'P4', 'F5', 'F6', 'F7', 'P8',
       'E9', 'E10', 'P11', 'F12', 'P13', 'F14', 'E15', 'F16', 'P17', 'E18',
       'F19', 'E20', 'E21', 'E22', 'E23', 'F24', 'P25',  'diagnosis']] #จัดเรียงข้อมูลใน DataFrame
```

```
#แยกข้อมูลออกเป็นข้อมูลที่ใช้ฝึกและผลลัพท์
Y=target = df.loc[:, 'diagnosis']
X=data = df[['age', 'P1', 'E2', 'F3', 'P4', 'F5', 'F6', 'F7', 'P8',
       'E9', 'E10', 'P11', 'F12', 'P13', 'F14', 'E15', 'F16', 'P17', 'E18',
       'F19', 'E20', 'E21', 'E22', 'E23', 'F24', 'P25']]
```

ภาพประกอบ 7 ตัวอย่างโค้ดการแปลงข้อมูลจากข้อความเป็นตัวเลข และแบ่งข้อมูลออกเป็นข้อมูลที่ใช้ฝึกและผลลัพท์

3.4.3 เลือกอัลกอริธึมสำหรับสร้างแบบจำลองทำนายข้อมูล โดยแต่ละอัลกอริธึมมีหลักการต่างกันเพื่อสามารถเปรียบเทียบแบบจำลองได้ ซึ่งอัลกอริธึมพื้นฐานที่ใช้ในงาน Machine Learning ประเภท Supervised learning สำหรับแก้ปัญหาเชิง Classification ได้แก่ K-Nearest Neighbors, Naïve Bayes, Support Vector Machine และ Decision trees [15] แต่เนื่องจากข้อมูลมีปริมาณไม่เยอะมาก จึงใช้ Random Forest แทนซึ่งเป็นหลักการคล้ายกันแต่ลดปัญหา Overfitting ได้ดีกว่า Decision Trees

3.4.4 สร้างแบบจำลองโดยใช้ฟีเจอร์ทั้งหมด ปรับพารามิเตอร์ และวัดประสิทธิภาพของแบบจำลอง

3.4.4.1 แบบจำลองสร้างโดยอัลกอริธึม Gaussian Naïve Bayes วัดประสิทธิภาพดังภาพประกอบ 8

```
gnb_clf = GaussianNB()
cv = 5 #กำหนดการแบ่งข้อมูลสำหรับทำ Cross Validation เพื่อทดสอบแบบจำลองเป็น 5 ส่วน
f1score = cross_val_score(gnb_clf,data,target, cv=cv,scoring='f1_macro') #วัดประสิทธิภาพของแบบจำลองโดยใช้ f1-score
accscore = cross_val_score(gnb_clf,data,target, cv=cv) #วัดประสิทธิภาพของแบบจำลองโดยใช้ accuracy score
recallscore = cross_val_score(gnb_clf,data,target, cv=cv,scoring='recall_macro') #วัดประสิทธิภาพของแบบจำลองโดยใช้ recall score
pscore = cross_val_score(gnb_clf,data,target, cv=cv,scoring='precision_macro') #วัดประสิทธิภาพของแบบจำลองโดยใช้ precision score
f_gnb = np.mean(f1score) #หาค่าเฉลี่ยของค่าประสิทธิภาพ f1-score ที่ได้ทั้งหมด 5 ครั้ง
acc_gnb = np.mean(accscore) #หาค่าเฉลี่ยของค่าประสิทธิภาพ accuracy score ที่ได้ทั้งหมด 5 ครั้ง
r_gnb = np.mean(recallscore) #หาค่าเฉลี่ยของค่าประสิทธิภาพ recall score ที่ได้ทั้งหมด 5 ครั้ง
p_gnb = np.mean(pscore) #หาค่าเฉลี่ยของค่าประสิทธิภาพ precision score ที่ได้ทั้งหมด 5 ครั้ง
print ('Accuracy:', acc_gnb)
print ('F1 score:', f_gnb)
print ('Recall:', r_gnb)
print ('Precision:', p_gnb)
```

ภาพประกอบ 8 ตัวอย่างโค้ดแสดงการวัดประสิทธิภาพแบบจำลองที่สร้างด้วยอัลกอริธึม GNB

3.4.4.2 แบบจำลองสร้างโดยอัลกอริธึม K-Nearest Neighbors ปรับพารามิเตอร์โดยใช้ GridSearchCV และวัดประสิทธิภาพดังภาพประกอบ 9

```
knn = KNeighborsClassifier()
k_range = list(range(1, 31))
p = [1,2]
weights=['distance','uniform']
param_grid = dict(n_neighbors=k_range,p=p,weights=weights)
grid = GridSearchCV(knn, param_grid, cv=5, scoring='accuracy')
grid.fit(X,Y)
print(grid.best_params_)
print(grid.best_estimator_)
```

```
clf_KNNd = KNeighborsClassifier(n_neighbors=21,p=1,weights='distance')
f1score = cross_val_score(clf_KNNd,data,target, cv=cv,scoring='f1_macro')
accscore = cross_val_score(clf_KNNd,data,target, cv=cv)
recallscore = cross_val_score(clf_KNNd,data,target, cv=cv,scoring='recall_macro')
pscore = cross_val_score(clf_KNNd,data,target, cv=cv,scoring='precision_macro')
f_knn = np.mean(f1score)
acc_knn = np.mean(accscore)
r_knn = np.mean(recallscore)
p_knn = np.mean(pscore)
print ('Accuracy:', acc_knn)
print ('F1 score:', f_knn)
print ('Recall:', r_knn)
print ('Precision:', p_knn)
```

ภาพประกอบ 9 ตัวอย่างโค้ดแสดงการปรับพารามิเตอร์ และวัดประสิทธิภาพแบบจำลองที่สร้างด้วยอัลกอริธึม KNN

3.4.4.3 แบบจำลองสร้างโดยอัลกอริธึม Support Vector Machine ปรับพารามิเตอร์โดยใช้ GridSearchCV และวัดประสิทธิภาพดังภาพประกอบ 10

```
C=range(1,10)
degree = range(1,3)
gamma = (0.1,5)
param_grid=dict(C=C,degree=degree,gamma=gamma)
grid=GridSearchCV(SVC( kernel='poly',decision_function_shape='ovr', random_state=42),param_grid,cv=5)
grid.fit(X, Y)

print(grid.best_params_)
print(grid.best_estimator_)
```

```
clf_SVC = SVC( C=1,degree=2,gamma=0.1,kernel='poly',decision_function_shape='ovr', random_state=42)
f1score = cross_val_score(clf_SVC,data,target, cv=cv,scoring='f1_macro')
accscore = cross_val_score(clf_SVC,data,target, cv=cv)
recallscore = cross_val_score(clf_SVC,data,target, cv=cv,scoring='recall_macro')
pscore = cross_val_score(clf_SVC,data,target, cv=cv,scoring='precision_macro')
f_svc = np.mean(f1score)
acc_svc = np.mean(accscore)
r_svc = np.mean(recallscore)
p_svc = np.mean(pscore)
print ('Accuracy:', acc_svc)
print ('F1 score:', f_svc)
print ('Recall:', r_svc)
print ('Precision:', p_svc)
```

ภาพประกอบ 10 ตัวอย่างโค้ดแสดงการปรับพารามิเตอร์ และวัดประสิทธิภาพแบบจำลองที่สร้างด้วยอัลกอริธึม SVM

3.4.4.4 แบบจำลองสร้างโดยอัลกอริธึม Random Forest ปรับพารามิเตอร์โดยใช้ RandomizedSearchCV และวัดประสิทธิภาพดังภาพประกอบ 11

```
n_estimators = list(range(2, 50))
max_features = ['auto', 'sqrt']
max_depth = list(range(2, 50))
min_samples_split = list(range(2, 20))
min_samples_leaf = list(range(2, 20))
random_grid = dict(n_estimators= n_estimators,
                   max_features = max_features,
                   max_depth= max_depth,
                   min_samples_split= min_samples_split,
                   min_samples_leaf= min_samples_leaf)
rf = RandomForestClassifier(random_state=42)
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid, n_iter = 50, cv = 5,
                               verbose=2, random_state=42, n_jobs = -1)
rf_random.fit(X, Y)
rf_random.best_params_
```

```
clf_RFC =  RandomForestClassifier(max_depth=6,max_features='auto',min_samples_leaf=10,
                                  min_samples_split=14,n_estimators=30,random_state=42)
accscore = cross_val_score(clf_RFC,data,target, cv=cv)
recallscore = cross_val_score(clf_RFC,data,target, cv=cv,scoring='recall_macro')
pscore = cross_val_score(clf_RFC,data,target, cv=cv,scoring='precision_macro')
F1_RFC = np.mean(f1score)
acc_RFC = np.mean(accscore)
recall_RFC = np.mean(recallscore)
p_RFC = np.mean(pscore)
print ('Accuracy:', acc_RFC)
print ('F1 score:', F1_RFC)
print ('Recall:', recall_RFC)
print ('Precision:', p_RFC)
```

ภาพประกอบ 11 ตัวอย่างโค้ดแสดงการปรับพารามิเตอร์ และวัดประสิทธิภาพแบบจำลองที่สร้างด้วยอัลกอริธึม RF

3.4.5 ศึกษาความสัมพันธ์ของฟีเจอร์ ดังภาพประกอบ 12

```
df2 = df[['age', 'P1', 'E2', 'F3', 'P4', 'F5', 'F6', 'F7', 'P8',
       'E9', 'E10', 'P11', 'F12', 'P13', 'F14', 'E15', 'F16', 'P17', 'E18',
       'F19', 'E20', 'E21', 'E22', 'E23', 'F24', 'P25', 'diagnosis']]#เรียงลำดับฟีเจอร์ตามแบบประเมินอาการวิงเวียนศีรษะ
fig, ax = plt.subplots(figsize=(20,20))
sns.heatmap(df2.corr(method='pearson'), annot=True, fmt=".2f",linewidths=.5, ax=ax) #สร้างกราฟแสดงค่าสัมประสิทธิ์สหสัมพันธ์แบบเพียร์สัน
```

ภาพประกอบ 12 ตัวอย่างโค้ดแสดงการสร้างกราฟแสดงค่าสัมประสิทธิ์สหสัมพันธ์แบบเพียร์สัน

3.4.6 สร้างฟีเจอร์ใหม่จากฟีเจอร์เดิมโดยใช้ค่าสัมประสิทธิ์สหสัมพันธ์แบบเพียร์สันของฟีเจอร์คัดเลือกฟีเจอร์ที่มีความสัมพันธ์กับประเภทของโรคเวียนศีรษะมากที่สุด และรวมฟีเจอร์เข้าด้วยกันดังภาพประกอบ 13 จากนั้นเลือกฟีเจอร์ที่เหมาะสมในการใช้ทำแบบจำลอง

```
F7_E23 = df['F7']+df['E23']
df['F7_E23'] = pd.Series(F7_E23)
```

ภาพประกอบ 13 โค้ดตัวอย่างการสร้างฟีเจอร์ใหม่คือ F7_E23

3.4.6.1 การคัดเลือกฟีเจอร์ด้วยวิธี Filter techniques โดยใช้ค่า Chi square ดังสมการ 8 และภาพประกอบ 14 และหาจำนวนฟีเจอร์ที่เหมาะสมเมื่อเรียงลำดับด้วยค่า Chi square ในการสร้างแบบจำลองด้วยอัลกอริธึมแต่ละอัลกอริธึม

$$X^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \quad (8)$$

X^2 = Chi square

O_i = ความถี่ที่ได้จากการสังเกต (Observed Frequency)

E_i = ความถี่ที่คาดหวัง (Expected Frequency)

```
y= df.loc[:, 'diagnosis']
X= df[['age', 'P1', 'E2', 'F3', 'P4', 'F5', 'F6', 'F7', 'P8',
       'E9', 'E10', 'P11', 'F12', 'P13', 'F14', 'E15', 'F16', 'P17', 'E18',
       'F19', 'E20', 'E21', 'E22', 'E23', 'F24', 'P25', 'F7_E23']]
test = SelectKBest(score_func=chi2, k=5)

names = ['age', 'P1', 'E2', 'F3', 'P4', 'F5', 'F6', 'F7', 'P8',
       'E9', 'E10', 'P11', 'F12', 'P13', 'F14', 'E15', 'F16', 'P17', 'E18',
       'F19', 'E20', 'E21', 'E22', 'E23', 'F24', 'P25', 'F7_E23']
feature = list()
chi = list()
for n in range(0, len(names)):
    test = SelectKBest(score_func=chi2, k=5)
    fit = test.fit(X,y)
    feature.append(names[n])
    chi.append(fit.scores_[n])

    print(names[n], fit.scores_[n])
```

ภาพประกอบ 14 ตัวอย่างโค้ดหาค่า Chi square ของแต่ละฟีเจอร์

งานวิจัยนี้ใช้เทคนิคการคัดเลือกฟีเจอร์ด้วย Chi square กับแบบจำลองที่สร้างด้วยอัลกอริธึมดังต่อไปนี้

- การหาจำนวนฟีเจอร์ที่เหมาะสมสำหรับสร้างแบบจำลองโดยใช้อัลกอริธึม Gaussian Naïve Bayes ดังภาพประกอบ 15

```
accGNB = list()
n_features = list()
#หา accuracy เมื่อเลือกจำนวนฟีเจอร์ที่แตกต่างกันตามลำดับค่า Chi square
for n in range(1,27):
    Xnew =  SelectKBest(chi2, k=n).fit_transform(X, y)
    x_train, x_test, y_train, y_test = train_test_split(Xnew, target, test_size=0.2, random_state=42)
    gnb_clf = GaussianNB()
    gnb_clf.fit(x_train, y_train)
    gnb_pred = gnb_clf.predict(x_test)
    accscore = cross_val_score(gnb_clf,Xnew,y, cv=5)
    acc_gnb = np.mean(accscore)
    accGNB.append(acc_gnb)
    n_features.append(n)

Chi2GNB=pd.DataFrame({"Number of Selected Features":n_features})
Chi2GNB['accuracy score']= accGNB #สร้าง DataFrame จำนวนฟีเจอร์และ accuracy ที่ได้
Chi2GNB.sort_values(by='Number of Selected Features', ascending=1) #เรียงลำดับตามจำนวนฟีเจอร์
```

```
#สร้างกราฟแสดงความสัมพันธ์ระหว่างจำนวนฟีเจอร์และ accuracy และหาจำนวนฟีเจอร์ที่ให้ค่า accuracy สูงสุด
NumberofSelectedFeatures = Chi2GNB['Number of Selected Features']
accuracy_score = Chi2GNB['accuracy score']
ax = plt.gca()
plt.title('Feature Selection (Chi-square) of Naive Bayes Model')
Chi2GNB.plot(kind='line',x='Number of Selected Features',y='accuracy score',ax=ax)
max_index = np.where(accuracy_score == max(accuracy_score))
accuracy_score_max = accuracy_score[max_index[0][0]]
NumberofSelectedFeatures_max = NumberofSelectedFeatures[max_index[0][0]]
maxValue = 'features ='+str(NumberofSelectedFeatures_max)+' , '+'Accuracy score ='+str(accuracy_score_max)
plt.annotate(maxValue, xy=(NumberofSelectedFeatures_max,accuracy_score_max))
plt.show()
```

ภาพประกอบ 15 ตัวอย่างโค้ดหาจำนวนฟีเจอร์ซึ่งจัดลำดับด้วยค่า Chi square สำหรับแบบจำลองสร้างโดยอัลกอริธึม GNB ที่ให้ค่า accuracy สูงสุด

- การหาจำนวนฟีเจอร์ที่เหมาะสมสำหรับสร้างแบบจำลองโดยใช้อัลกอริธึม K-Nearest Neighbors ดังภาพประกอบ 16

```
n_features = list()
accKNN = list()
for n in range(1,27):
    Xnew =  SelectKBest(chi2, k=n).fit_transform(X, y)
    x_train, x_test, y_train, y_test = train_test_split(Xnew, target, test_size=0.2, random_state=42)
    knn_clf = KNeighborsClassifier()
    knn_clf.fit(x_train, y_train)
    knn_pred = knn_clf.predict(x_test)
    accscore = cross_val_score(knn_clf,Xnew,y, cv=5)
    acc_knn = np.mean(accscore)
    accKNN.append(acc_knn)
    n_features.append(n)
```

```
Chi2KNN=pd.DataFrame({"Number of Selected Features":n_features})

Chi2KNN['accuracy score']= accKNN
Chi2KNN.sort_values(by='Number of Selected Features', ascending=1)
NumberofSelectedFeatures = Chi2KNN['Number of Selected Features']
accuracy_score = Chi2KNN['accuracy score']
ax = plt.gca()
max_index = np.where(accuracy_score == max(accuracy_score))
accuracy_score_max = accuracy_score[max_index[0][0]]
NumberofSelectedFeatures_max = NumberofSelectedFeatures[max_index[0][0]]
maxValue = 'features = '+str(NumberofSelectedFeatures_max)+' , '+'Accuracy = '+str(accuracy_score_max)
plt.annotate(maxValue, xy=(NumberofSelectedFeatures_max,accuracy_score_max))
plt.title('Feature Selection (Chi-square) of KNN Model')
Chi2KNN.plot(kind='line',x='Number of Selected Features',y='accuracy score',ax=ax)
plt.show()
```

ภาพประกอบ 16 ตัวอย่างโค้ดหาจำนวนฟีเจอร์ซึ่งจัดลำดับด้วยค่า Chi square สำหรับแบบจำลองสร้างโดยอัลกอริธึม KNN ที่ให้ค่า accuracy สูงสุด

- การหาจำนวนฟีเจอร์ที่เหมาะสมสำหรับสร้างแบบจำลองโดยใช้อัลกอริธึม Support Vector Machine ดังภาพประกอบ 17

```
n_features = list()
accSVC = list()
for n in range(1,27):
    Xnew =  SelectKBest(chi2, k=n).fit_transform(X, y)
    svc_clf = SVC( random_state=42)
    accscore = cross_val_score(svc_clf,Xnew,y, cv=5)
    acc_svc = np.mean(accscore)
    accSVC.append(acc_svc)
    n_features.append(n)
```

```
Chi2SVC=pd.DataFrame({"Number of Selected Features":n_features})

Chi2SVC['accuracy score']= accSVC
Chi2SVC.sort_values(by='Number of Selected Features', ascending=1)

NumberofSelectedFeatures = Chi2SVC['Number of Selected Features']
accuracy_score = Chi2SVC['accuracy score']
ax = plt.gca()
max_index = np.where(accuracy_score == max(accuracy_score))
accuracy_score_max = accuracy_score[max_index[0][0]]
NumberofSelectedFeatures_max = NumberofSelectedFeatures[max_index[0][0]]
maxValue = 'features = '+str(NumberofSelectedFeatures_max)+' , '+'Accuracy = '+str(accuracy_score_max)
plt.annotate(maxValue, xy=(NumberofSelectedFeatures_max,accuracy_score_max))

ax = plt.gca()
plt.title('Feature Selection (Chi-square) of SVM Model')
Chi2SVC.plot(kind='line',x='Number of Selected Features',y='accuracy score',ax=ax)
plt.show()
```

ภาพประกอบ *17* ตัวอย่างโค้ดหาจำนวนฟีเจอร์ซึ่งจัดลำดับด้วยค่า Chi square สำหรับแบบจำลองสร้างโดยอัลกอริธึม SVM ที่ให้ค่า accuracy สูงสุด

- การหาจำนวนฟีเจอร์ที่เหมาะสมสำหรับสร้างแบบจำลองโดยใช้อัลกอริธึม Random Forest ดังภาพประกอบ 18

```
n_features = list()
accRF = list()
for n in range(1,27):
    Xnew =  SelectKBest(chi2, k=n).fit_transform(X, y)
    rf_clf = RandomForestClassifier(random_state=42)
    accscore = cross_val_score(rf_clf,Xnew,y, cv=5)
    acc_rf = np.mean(accscore)
    accRF.append(acc_rf)
    n_features.append(n)
```

```
Chi2RF=pd.DataFrame({"Number of Selected Features":n_features})

Chi2RF['accuracy score']= accRF
Chi2RF.sort_values(by='Number of Selected Features', ascending=1)

NumberofSelectedFeatures = Chi2RF['Number of Selected Features']
accuracy_score = Chi2RF['accuracy score']
ax = plt.gca()
max_index = np.where(accuracy_score == max(accuracy_score))
accuracy_score_max = accuracy_score[max_index[0][0]]
NumberofSelectedFeatures_max = NumberofSelectedFeatures[max_index[0][0]]
maxValue = 'features = '+str(NumberofSelectedFeatures_max)+' , '+'Accuracy = '+str(accuracy_score_max)
plt.annotate(maxValue, xy=(NumberofSelectedFeatures_max,accuracy_score_max))
ax = plt.gca()
plt.title('Feature Selection (Chi-square) of RF Model')
Chi2RF.plot(kind='line',x='Number of Selected Features',y='accuracy score',ax=ax)
plt.show()
```

ภาพประกอบ 18 ตัวอย่างโค้ดหาจำนวนฟีเจอร์ซึ่งจัดลำดับด้วยค่า Chi square สำหรับแบบจำลองสร้างโดยอัลกอริธึม RF ที่ให้ค่า accuracy สูงสุด

3.4.6.2 การคัดเลือกฟีเจอร์ด้วยวิธี Wrapper techniques โดยใช้เทคนิค Recursive Feature Elimination (RFE) กับแบบจำลองที่ใช้อัลกอริธึมดังต่อไปนี้

- การหาจำนวนฟีเจอร์ที่เหมาะสมโดยเทคนิค RFE กับแบบจำลองที่สร้างด้วยอัลกอริธึม Support Vector Machine โดยใช้ค่า coefficient ของฟีเจอร์เพื่อเรียงลำดับความสำคัญของฟีเจอร์ ดังภาพประกอบที่ 19

```
viz = RFECV(SVC(kernel='linear', random_state = 42), cv=5, scoring='accuracy')
viz.fit(data,target)

viz.poof()
```

```
ranking_ds=pd.DataFrame({"Features":X.columns})
fit=viz.fit(data,target)
ref_ranking=viz.ranking_
ranking_ds['Ranking']=ref_ranking
ranking_ds = ranking_ds.sort_values(by='Ranking', ascending=1)
ranking_ds
```

ภาพประกอบ 19 ตัวอย่างโค้ดการหาจำนวนฟีเจอร์ที่เหมาะสมสำหรับแบบจำลองสร้างโดยอัลกอริธึม SVM โดยใช้เทคนิค RFE

- การหาจำนวนฟีเจอร์ที่เหมาะสมโดยเทคนิค RFE กับแบบจำลองที่สร้างด้วยอัลกอริธึม Random Forest โดยใช้ค่า Feature importance ของฟีเจอร์เพื่อเรียงลำดับความสำคัญของฟีเจอร์ ดังภาพประกอบที่ 20

```
y=target = df.loc[:, 'diagnosis']
X=data = df[['age', 'P1', 'E2', 'F3', 'P4', 'F5', 'F6', 'F7', 'P8',
       'E9', 'E10', 'P11', 'F12', 'P13', 'F14', 'E15', 'F16', 'P17', 'E18',
       'F19', 'E20', 'E21', 'E22', 'E23', 'F24', 'P25', 'F7_E23']]
cv = StratifiedKFold(5)
rfe = RFECV(RandomForestClassifier(random_state=42), cv=cv, scoring='accuracy')

rfe.fit(X,y)
rfe.poof()
```

```
ranking_ds=pd.DataFrame({"Features":X.columns})
fit=rfe.fit(X,y)
ref_ranking=rfe.ranking_
ranking_ds['Ranking']=ref_ranking
ranking_ds = ranking_ds.sort_values(by='Ranking', ascending=1)
ranking_ds
```

ภาพประกอบ 20 ตัวอย่างโค้ดการหาจำนวนฟีเจอร์ที่เหมาะสมสำหรับแบบจำลองสร้างโดยอัลกอริธึม RF โดยใช้เทคนิค RFE

3.4.6.3 การสกัดฟีเจอร์เพื่อลดมิติของฟีเจอร์ด้วยวิธี Principal Component Analysis (PCA) กับแบบจำลองที่ใช้อัลกอริธึมดังต่อไปนี้

- การหาองค์ประกอบที่ให้ Accuracy ดีที่สุดในการปรับมิติฟีเจอร์ด้วย PCA กับแบบจำลองที่สร้างด้วยอัลกอริธึม Gaussian Naïve Bayes ดังภาพประกอบ 21

```
accGNB = list()
for n in range(1,27):
    Xnew =  PCA(n_components=n,random_state = 42).fit_transform(X, y)
    gnb_clf = GaussianNB()
    accscore = cross_val_score(gnb_clf,Xnew,y, cv=5)
    acc_gnb = np.mean(accscore)
    accGNB.append(acc_gnb)

PCAGNB=pd.DataFrame({"Number of Components to Keep":n_component})
PCAGNB['accuracy score']= accGNB
PCAGNB.sort_values(by='Number of Components to Keep', ascending=1)
n_components = PCAGNB['Number of Components to Keep']
accuracy_score = PCAGNB['accuracy score']
ax = plt.gca()
max_index = np.where(accuracy_score == max(accuracy_score))
accuracy_score_max = accuracy_score[max_index[0][0]]
n_components_max = n_components[max_index[0][0]]
maxValue = 'n_components = '+str(n_components_max)+' , '+'Accuracy = '+str(accuracy_score_max)
plt.annotate(maxValue, xy=(n_components_max-6,accuracy_score_max-0.005))
PCAGNB.plot(kind='line',x='Number of Components to Keep',y='accuracy score',ax=ax)
plt.show()
```

ภาพประกอบ 21 ตัวอย่างโค้ดหาองค์ประกอบที่ให้ accuracy สูงสุดในการทำ PCA เพื่อสร้างแบบจำลองโดยใช้อัลกอริธึม Naïve Bayes

- การหาองค์ประกอบที่ให้ Accuracy ดีที่สุดในการปรับมิติฟีเจอร์ด้วย PCA กับแบบจำลองที่สร้างด้วยอัลกอริธึม K-Nearest Neighbors ดังภาพประกอบ 22

```
y=target = df.loc[:, 'diagnosis']
X=data = df[['age', 'P1', 'E2', 'F3', 'P4', 'F5', 'F6', 'F7', 'P8',
       'E9', 'E10', 'P11', 'F12', 'P13', 'F14', 'E15', 'F16', 'P17', 'E18',
       'F19', 'E20', 'E21', 'E22', 'E23', 'F24', 'P25', 'F7_E23']]
n_component = list()
accKNN = list()
for n in range(1,27):
    Xnew =  PCA(n_components=n,random_state = 42).fit_transform(X, y)
    knn_clf = KNeighborsClassifier()
    accscore = cross_val_score(knn_clf,Xnew,y, cv=5)
    acc_knn = np.mean(accscore)
    accKNN.append(acc_knn)
    n_component.append(n)
```

```
PCAKNN=pd.DataFrame({"Number of Components to Keep":n_component})
PCAKNN['accuracy score']= accKNN
PCAKNN.sort_values(by='Number of Components to Keep', ascending=1)
n_components = PCAKNN['Number of Components to Keep']
accuracy_score = PCAKNN['accuracy score']
ax = plt.gca()
max_index = np.where(accuracy_score == max(accuracy_score))
accuracy_score_max = accuracy_score[max_index[0][0]]
n_components_max = n_components[max_index[0][0]]
maxValue = 'n_components = '+str(n_components_max)+' , '+'Accuracy = '+str(accuracy_score_max)
plt.annotate(maxValue, xy=(n_components_max-2,accuracy_score_max))
ax = plt.gca()
PCAKNN.plot(kind='line',x='Number of Components to Keep',y='accuracy score',ax=ax)
plt.show()
```

ภาพประกอบ 22 ตัวอย่างโค้ดหาองค์ประกอบที่ให้ accuracy สูงสุดในการทำ PCA เพื่อสร้างแบบจำลองโดยใช้อัลกอริธึม KNN

- การหาองค์ประกอบที่ให้ Accuracy ดีที่สุดในการปรับมิติฟีเจอร์ด้วย PCA กับแบบจำลองที่สร้างด้วยอัลกอริธึม Support Vector Machine ดังภาพประกอบ 23

```
accSVC = list()
for n in range(1,27):
    Xnew =  PCA(n_components=n,random_state = 42).fit_transform(X, y)
    svc_clf = SVC(random_state=42)
    accscore = cross_val_score(svc_clf,Xnew,y, cv=5)
    acc_svc = np.mean(accscore)
    accSVC.append(acc_svc)

PCASVC=pd.DataFrame({"Number of Components to Keep":n_component})
PCASVC['accuracy score']= accSVC
PCASVC.sort_values(by='Number of Components to Keep', ascending=1)
n_components = PCASVC['Number of Components to Keep']
accuracy_score = PCASVC['accuracy score']
ax = plt.gca()
max_index = np.where(accuracy_score == max(accuracy_score))
accuracy_score_max = accuracy_score[max_index[0][0]]
n_components_max = n_components[max_index[0][0]]
maxValue = 'n_components = '+str(n_components_max)+' , '+'Accuracy = '+str(accuracy_score_max)
plt.annotate(maxValue, xy=(n_components_max-6,accuracy_score_max-0.005))
PCASVC.plot(kind='line',x='Number of Components to Keep',y='accuracy score',ax=ax)
plt.show()
```

ภาพประกอบ 23 ตัวอย่างโค้ดหาองค์ประกอบที่ให้ accuracy สูงสุดในการทำ PCA เพื่อสร้างแบบจำลองโดยใช้อัลกอริธึม SVM

- การหาองค์ประกอบที่ให้ Accuracy ดีที่สุดในการปรับมิติฟีเจอร์ด้วย PCA กับแบบจำลองที่สร้างด้วยอัลกอริธึม Random Forest ดังภาพประกอบ 24

```
accRF = list()
for n in range(1,27):
    Xnew =  PCA(n_components=n,random_state = 42).fit_transform(X, y)
    rf_clf =  RandomForestClassifier(random_state=42)
    accscore = cross_val_score(rf_clf,Xnew,y, cv=5)
    acc_rf = np.mean(accscore)
    accRF.append(acc_rf)

PCARF=pd.DataFrame({"Number of Components to Keep":n_component})
PCARF['accuracy score']= accRF
PCARF.sort_values(by='Number of Components to Keep', ascending=1)
n_components = PCARF['Number of Components to Keep']
accuracy_score = PCARF['accuracy score']
ax = plt.gca()
max_index = np.where(accuracy_score == max(accuracy_score))
accuracy_score_max = accuracy_score[max_index[0][0]]
n_components_max = n_components[max_index[0][0]]
maxValue = 'n_components = '+str(n_components_max)+' , '+'Accuracy = '+str(accuracy_score_max)
plt.annotate(maxValue, xy=(n_components_max-6,accuracy_score_max))
PCARF.plot(kind='line',x='Number of Components to Keep',y='accuracy score',ax=ax)
plt.show()
```

ภาพประกอบ 24 ตัวอย่างโค้ดหาองค์ประกอบที่ให้ accuracy สูงสุดในการทำ PCA เพื่อสร้างแบบจำลองโดยใช้อัลกอริธึม RF

3.4.7 เปรียบเทียบประสิทธิภาพของแบบจำลองที่สร้างด้วยอัลกอริธึมต่างกันและฟีเจอร์ที่แตกต่างกันเพื่อหาแบบจำลองที่ให้ประสิทธิภาพที่ดีที่สุด

3.4.8 เครื่องมือที่ใช้วัดประสิทธิภาพของแบบจำลอง

3.4.8.1 Precision Score (สมการ 9)

$$Precision\ score = \frac{TP}{TP+FP} \quad (9)$$

TP = True Positive

FP = False Positive

3.4.8.2 Accuracy score (สมการ 10)

$$Accuracy\ score = \frac{TP+TN}{TP+FN+TN+FP} \quad (10)$$

FN = False Negative

TN = True Negative

3.4.8.3 Recall score (สมการ 11)

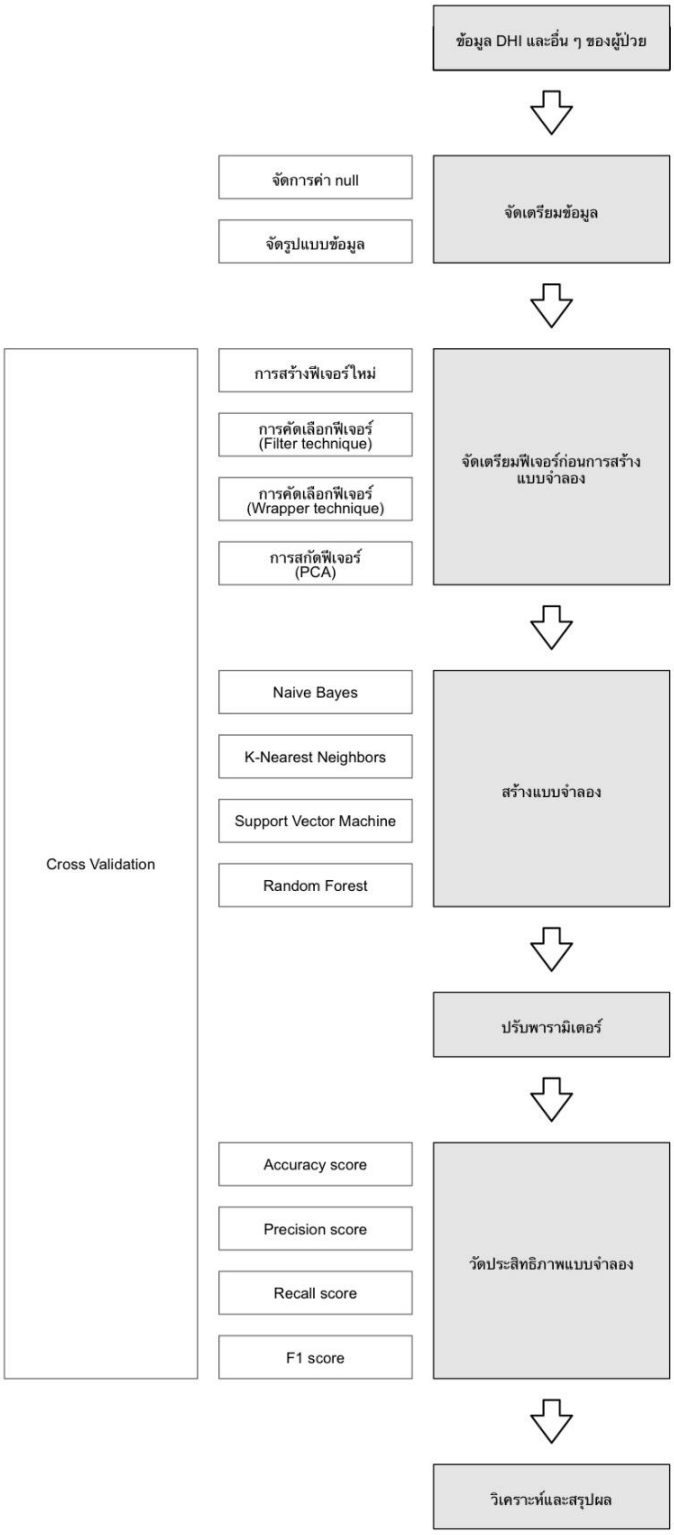
$$Recall\ score = \frac{TP}{TP+FN} \quad (11)$$

3.4.8.4 F1-score (สมการ 12)

$$F1\ score = \frac{2\times Precision\ score\times Recall\ score}{Precision\ score+Recall\ score} \quad (12)$$

3.5 แผนผังแสดงลำดับขั้นตอนการดำเนินการวิจัย

การทำวิจัยเรื่องการจำแนกประเภทของโรคเวียนศีรษะขณะเปลี่ยนท่าจากข้อมูลแบบประเมินอาการวิงเวียนศีรษะโดยใช้เทคนิคการเรียนรู้ของเครื่องสามารถแสดงแผนผังลำดับขั้นตอนการดำเนินการได้ดังภาพประกอบที่ 25



ภาพประกอบ 25 แผนผังแสดงลำดับการดำเนินการวิจัย

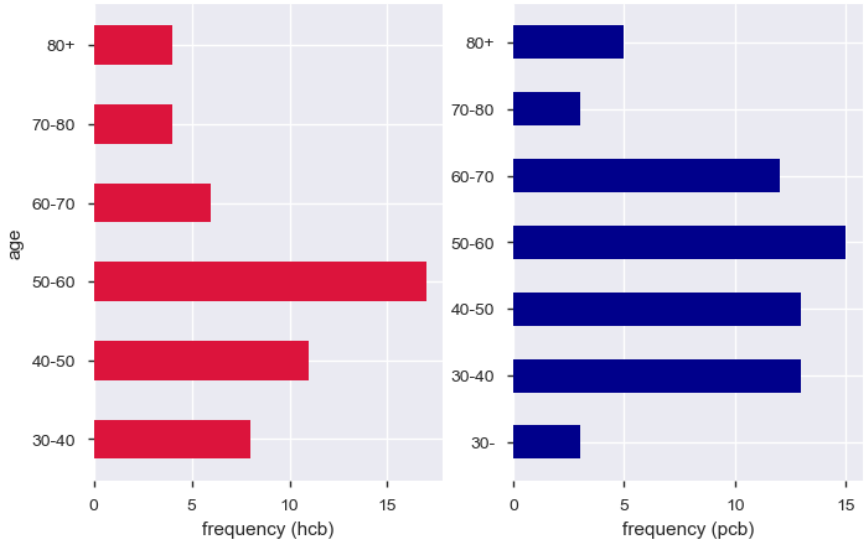
บทที่ 4
ผลการดำเนินการวิจัย

ในการวิจัยเรื่องการจำแนกประเภทโรคเวียนศีรษะขณะเปลี่ยนท่าจากข้อมูลแบบประเมินอาการวิงเวียนศีรษะโดยใช้เทคนิคการเรียนรู้ของเครื่อง ผู้วิจัยได้ดำเนินการวิจัยโดยการศึกษาตามขั้นตอนต่าง ๆ ตลอดจนการวัดประสิทธิภาพ เพื่อให้บรรลุจุดประสงค์ของการวิจัยที่ได้กำหนดไว้ ได้ดังนี้

1. ผลลัพธ์ของการวิเคราะห์ข้อมูล
2. ผลลัพธ์ของการสร้างแบบจำลองโดยใช้ข้อมูลจากแบบประเมินอาการวิงเวียนศีรษะ
3. ผลลัพธ์ของการวิเคราะห์ฟีเจอร์
4. ผลลัพธ์ของการสร้างแบบจำลองโดยใช้ฟีเจอร์ที่ได้จากการคัดเลือกด้วยวิธี chi-square
5. ผลลัพธ์ของการสร้างแบบจำลองโดยใช้ฟีเจอร์ที่ได้จากการคัดเลือกด้วยวิธี Recursive Feature Elimination
6. ผลลัพธ์ของการสร้างแบบจำลองโดยใช้ฟีเจอร์ที่ได้จากการทำ Principal Component Analysis

4.1 ผลลัพธ์ของการวิเคราะห์ข้อมูล

การวิจัยนี้ มีผู้ที่เป็นโรคเวียนศีรษะขณะเปลี่ยนท่า (Benign Paroxysmal Positioning Vertigo) ทั้งหมด 114 ราย โดยสามารถแบ่งได้ 2 ประเภท ประกอบด้วย โรคตะกอนหินปูนในหูชั้นในหลุดชนิดโพสทีเรียร์แคแนล หรือ Posterior canal - Benign Paroxysmal Positioning Vertigo (PC-BPPV) 64 รายหรือคิดเป็น 56.1% และ โรคตะกอนหินปูนในหูชั้นในหลุดชนิดฮอริซอนทอลแคแนล หรือ Horizontal canal - Benign Paroxysmal Positioning Vertigo (HC-BPPV) 50 รายหรือคิดเป็น 43.9% โดยพบว่าโรคเวียนศีรษะขณะเปลี่ยนท่าพบได้ในผู้ป่วยอายุตั้งแต่ 20 ปี ถึง 84 ปี โดยมีอายุเฉลี่ย 52.64 ปีซึ่งใกล้เคียงกับอายุเฉลี่ยของผู้ป่วย BPPV ในเยอรมนี [11] ซึ่งโรคเวียนศีรษะทั้ง 2 ชนิดส่วนใหญ่พบในช่วงอายุ 50 – 60 ปี ดังที่แสดงในภาพประกอบ 26



ภาพประกอบ 26 กราฟแจกแจงความถี่ช่วงอายุของโรคเวียนศีรษะขณะเปลี่ยนท่า ชนิดฮอริซอนทอลแคแนล และโพสทีเรียร์แคแนล

4.2 ผลลัพธ์ของการสร้างแบบจำลองโดยใช้ข้อมูลจากแบบประเมินอาการวิงเวียนศีรษะ

เมื่อสร้างแบบจำลองจำแนกประเภทโรควิงเวียนศีรษะขณะเปลี่ยนท่าโดยกำหนดให้ฟีเจอร์จำนวน 26 ฟีเจอร์ ได้แก่ age, P1, E2, F3, P4, F5, F6, F7, P8, E9, E10, P11, F12, P13, F14, E15, F16, P17, E18, F19, E20, E21, E22, E23, F24, P25 โดยกำหนดให้ diagnosis หรือประเภทของโรควิงเวียนศีรษะเป็นฟีเจอร์ประเภทลาเบล และแบ่งข้อมูลทั้งหมดเป็น 2 ส่วนคือ 80% ของข้อมูลทั้งหมดเป็นข้อมูลที่ใช้ในการสร้างแบบจำลอง (training set) และ 20 % ของข้อมูลทั้งหมดเป็นข้อมูลที่ใช้ในการวัดประสิทธิภาพของแบบจำลอง (testing set) โดยสร้างแบบจำลองด้วยอัลกอริธึมทั้งหมด 4 อัลกอริธึม ได้แก่ Random forest, Naïve Bayes, SVM และ KNN และหาพารามิเตอร์ของแต่ละแบบจำลองที่ให้ประสิทธิภาพที่ดีที่สุดโดยใช้เครื่องมือ GridSearchCV และ RandomizedSearchCV โดยมีรายละเอียดของพารามิเตอร์และประสิทธิภาพของแต่ละแบบจำลองดังตาราง 2

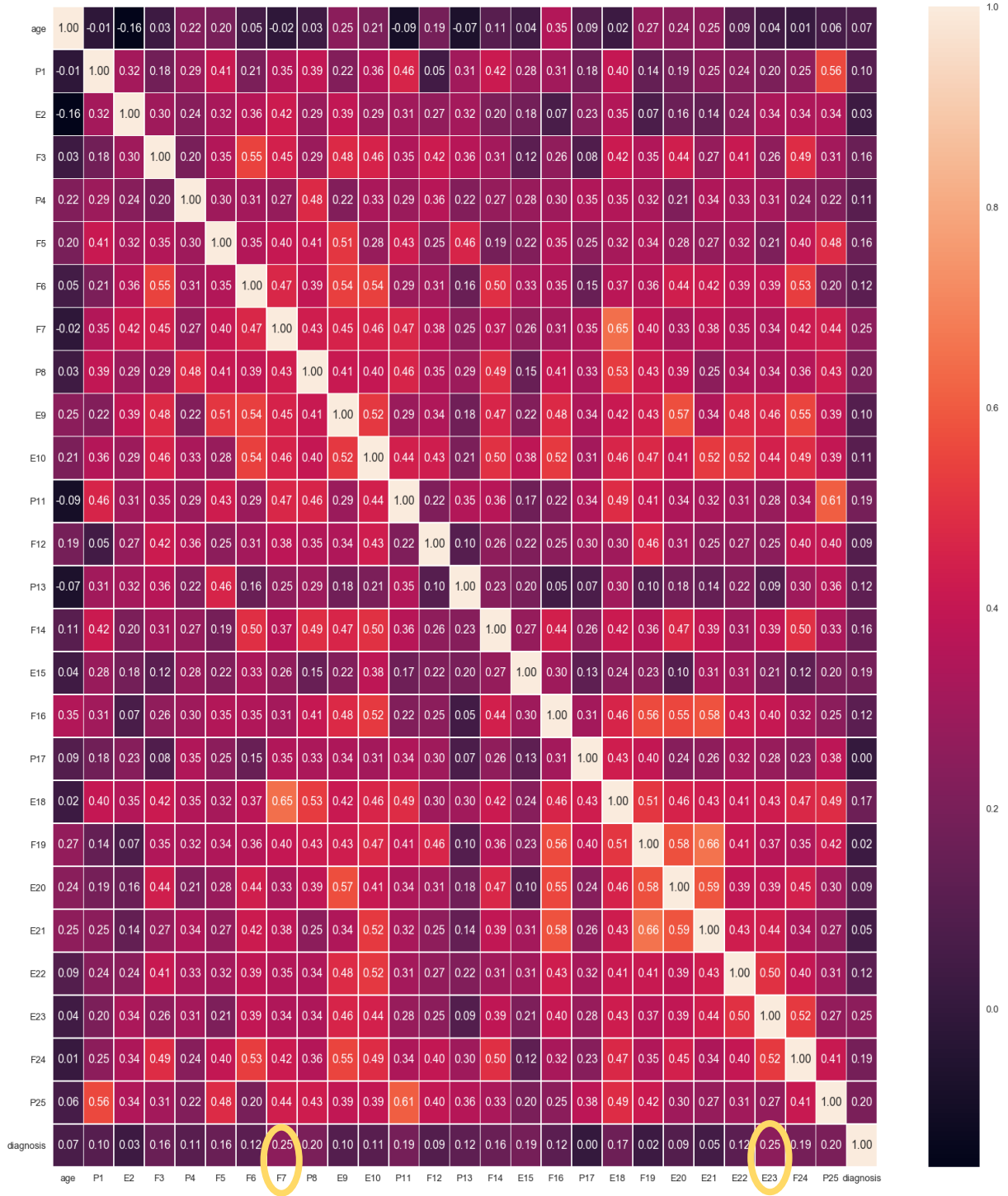
ตาราง 2 พารามิเตอร์และประสิทธิภาพของแบบจำลองจำแนกประเภทโรควิงเวียนศีรษะขณะเปลี่ยนท่าแต่ละอัลกอริธึม โดยใช้ฟีเจอร์จากแบบประเมินอาการวิงเวียนศีรษะและอายุของผู้ป่วย

Algorithm	parameter	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Gaussian Naive Bayes	priors=None	54.27	53.41	53.61	53.35
K-Nearest Neighbors	n_neighbors=21, p=1, weights='distance'	52.61	50.44	51.18	49.69
Support Vector Classification	C=1, degree=2, gamma=0.1, kernel='poly'	57.04	56.59	56.12	55.32
Random Forest	n_estimators=30, max_depth=6, max_features= 'auto', min_samples_split=10, min_samples_leaf=14	57.87	60.04	54.18	55.31

ผลจากตาราง 2 แสดงให้เห็นว่าประสิทธิภาพของการทำนายประเภทของโรงเวียนศีรษะขณะเปลี่ยนท่าในแต่ละแบบจำลองยังไม่ดีพอ การหาฟีเจอร์ที่เหมาะสมและการคัดเลือกฟีเจอร์เป็นวิธีที่ช่วยพัฒนาประสิทธิภาพของแบบจำลองได้ ดังนั้นการหาความสัมพันธ์ระหว่างฟีเจอร์จึงเป็นขั้นตอนถัดไปในการดำเนินการวิจัยเพื่อช่วยในการหาฟีเจอร์ที่เหมาะสมยิ่งขึ้น

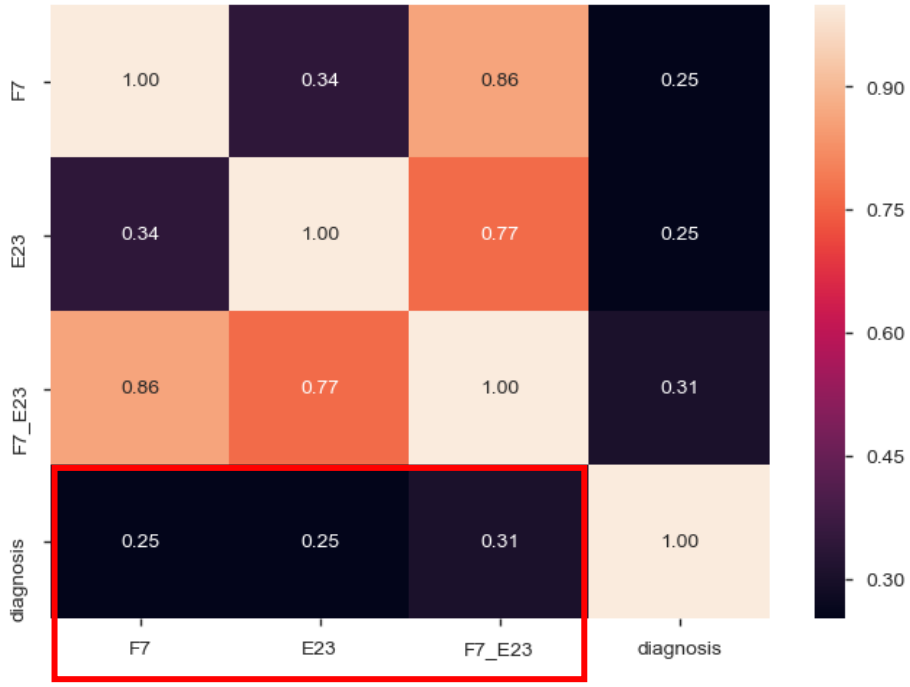
4.3 ผลลัพธ์ของการวิเคราะห์ฟีเจอร์สำหรับสร้างฟีเจอร์ (feature engineering)

ฟีเจอร์ที่ใช้ในชุดข้อมูลแบบประเมินอาการวิงเวียนศีรษะ (Dizziness Handicap Inventory) มีทั้งหมด 26 ฟีเจอร์ ประกอบด้วยคะแนนจากแบบทดสอบแต่ละข้อทั้งหมด 25 ข้อ อายุของผู้ป่วย และประเภทของโรคเวียนศีรษะขณะเปลี่ยนท่า โดยให้คะแนนจากแบบทดสอบแต่ละข้อทั้งหมด 25 ข้อ และอายุของผู้ป่วยเป็นตัวแปรต้น และให้ประเภทของโรคเวียนศีรษะขณะเปลี่ยนท่าเป็นตัวแปรตาม จากนั้นหาความสัมพันธ์ของแต่ละตัวแปรโดยใช้สัมประสิทธิ์สหสัมพันธ์แบบเพียร์สัน (Pearson’s Correlation) ได้ดังภาพประกอบ 27



ภาพประกอบ 27 ค่าสัมประสิทธิ์สหสัมพันธ์แบบเพียร์สันของฟีเจอร์ที่ใช้ในชุดข้อมูลแบบประเมินอาการวิงเวียนศีรษะ

เมื่อพิจารณาค่าสัมประสิทธิ์สหสัมพันธ์แบบเพียร์สันของแต่ละตัวแปรต้นกับตัวแปรตามแล้ว พบว่าตัวแปรต้นที่มีค่าสัมประสิทธิ์สหสัมพันธ์สูงสุดคือ คะแนนจากข้อที่ 7 (F7) และ ข้อที่ 23 (E23) ของชุดข้อมูลประเมินอาการวิงเวียนศีรษะ คือ มีค่าสัมประสิทธิ์สหสัมพันธ์ 0.25 ดังนั้นหากนำตัวแปรทั้งสองมาสร้างตัวแปรใหม่ น่าจะมีความสัมพันธ์กับตัวแปรตามมากขึ้นและได้ตัวแปรที่เป็นฟีเจอร์สำคัญที่จะช่วยพัฒนาแบบจำลองให้มีประสิทธิภาพมากขึ้นได้ หลังจากสร้างตัวแปรใหม่ คือ ตัวแปร F7_E23 ซึ่งเกิดจากผลรวมของคะแนนจากข้อที่ 7 (F7) และ คะแนนจากข้อที่ 23 (E23) เมื่อคำนวณค่าสัมประสิทธิ์สหสัมพันธ์ของตัวแปร F7_E23 กับตัวแปรตาม พบว่าได้ค่าสัมประสิทธิ์สหสัมพันธ์ 0.31 ซึ่งมากกว่าค่าสัมประสิทธิ์สหสัมพันธ์ของตัวแปรต้นอื่น ๆ กับตัวแปรตาม (ภาพประกอบ 28) ดังนั้นจึงนำ F7_E23 มาใช้ในการสร้างแบบจำลองเพื่อเปรียบเทียบประสิทธิภาพของแบบจำลอง



ภาพประกอบ 28 แสดงค่าสัมประสิทธิ์สหสัมพันธ์แบบเพียร์สันของ F7_E23 เทียบกับ F7 และ E23

4.4 ผลลัพธ์ของการสร้างแบบจำลองโดยใช้ฟีเจอร์ที่ได้จากการคัดเลือกด้วยวิธี chi-square

ในการคัดเลือกฟีเจอร์ด้วยวิธี chi-square ผู้วิจัยได้หาค่า chi-square ของแต่ละฟีเจอร์ ดังตาราง 3 เพื่อคัดเลือกฟีเจอร์ที่มีค่า chi-square สูงมาใช้สร้างแบบจำลองในแต่ละอัลกอริธึม โดยค่า chi-square จะแสดงถึงความเป็นอิสระระหว่างฟีเจอร์กับประเภทของโรคเวียนศีรษะขณะเปลี่ยนท่า ฟีเจอร์ที่มีค่า chi-square ที่สูงแสดงว่าฟีเจอร์นั้นมีความเป็นอิสระกับประเภทของโรคเวียนศีรษะ

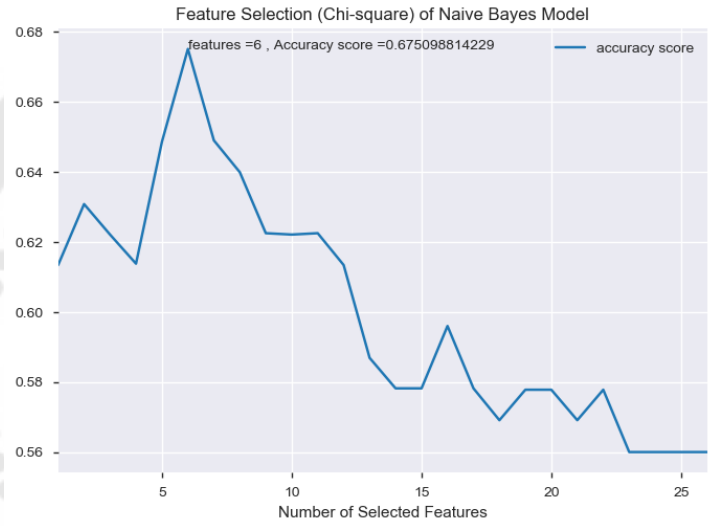
ขณะเปลี่ยนท่าน้อย และเป็นฟีเจอร์ที่มีความสำคัญต่อการจำแนกประเภทโรคเวียนศีรษะขณะเปลี่ยนท่ามาก

ตาราง 3 ค่า chi-square ของฟีเจอร์จากแบบทดสอบอาการวิงเวียนศีรษะ อายุ และ F7_E23

ลำดับ	ฟีเจอร์	chi-square
1	F7_E23	17.3165405405000
2	E15	9.48892857143000
3	E23	8.98565217391000
4	F7	8.66284482759000
5	P8	6.08950495050000
6	F24	4.38020833330000
7	E18	3.90328804348000
8	E22	2.97037500000000
9	P11	2.81535000000000
10	P4	2.74620192308000
11	E10	2.74620192308000
12	F16	2.70172535211000
13	F3	2.68508474576000
14	age	2.27525099983000
15	F6	1.96765957447000
16	E20	1.73069767442000
17	F5	1.41323529412000
18	E9	1.31011467890000
19	F12	1.19019396552000
20	P13	1.13645569620000
21	P1	0.94531250000000
22	E21	0.58782608695700
23	F19	0.13230000000000
24	E2	0.08978000000000
25	P17	0.00207627118644

4.4.1 แบบจำลองที่สร้างด้วยอัลกอริธึม Naïve Bayes

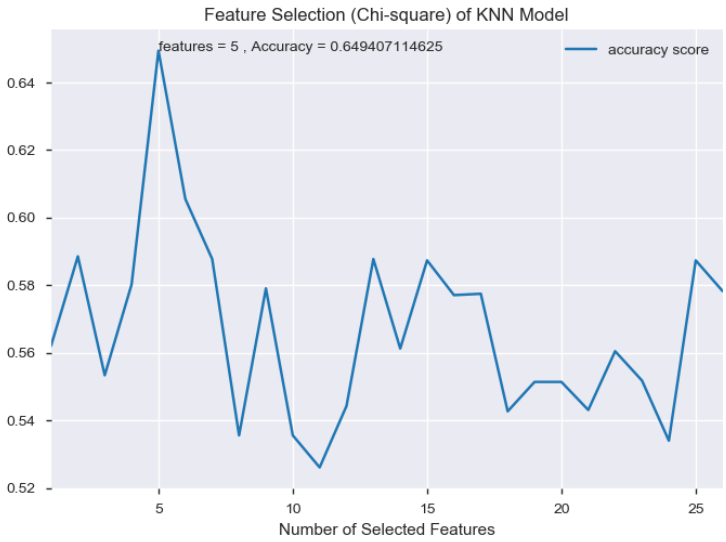
การสร้างแบบจำลองด้วยอัลกอริธึม Naïve Bayes นั้น ได้ใช้ SelectKBest เพื่อหาจำนวนฟีเจอร์ที่มีค่า chi-square มากที่สุดมาใช้ในการสร้างแบบจำลอง โดยพบว่าที่จำนวนฟีเจอร์เท่ากับ 6 ให้ค่า accuracy สูงที่สุด ดังภาพประกอบ 29 จึงเลือกฟีเจอร์ที่มี chi-square สูงสุด 6 ลำดับ ได้แก่ F7_E23, E15, E23, F7, P8 และ F24 ในการสร้างแบบจำลองด้วยอัลกอริธึม Naïve Bayes และให้ค่าประสิทธิภาพ ดังนี้ Accuracy 0.6751, F1 score 0.6595, Recall 0.6597 และ Precision 0.6720



ภาพประกอบ 29 จำนวนฟีเจอร์และค่า accuracy เมื่อใช้ฟีเจอร์ที่เรียงลำดับด้วย Chi-Square ในการสร้างแบบจำลองด้วยอัลกอริธึม Naive Bayes

4.4.2 แบบจำลองที่สร้างด้วยอัลกอริธึม K Nearest Neighbors (KNN)

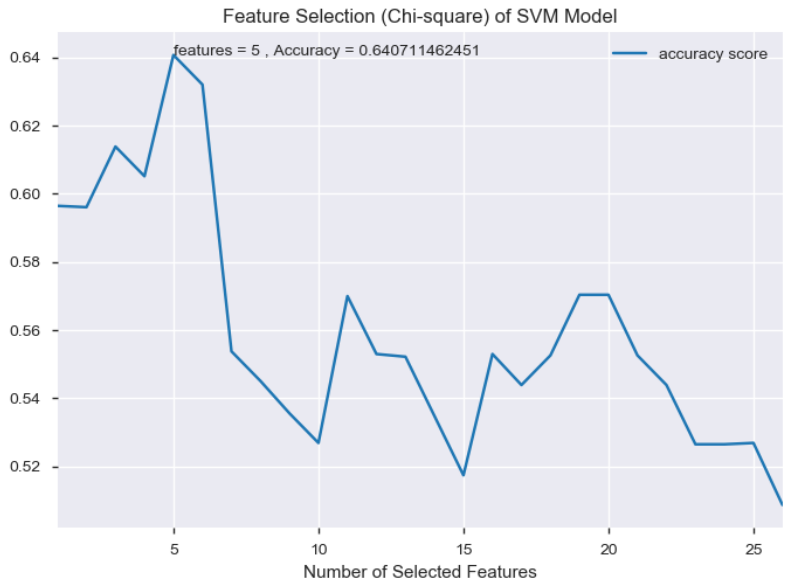
การสร้างแบบจำลองด้วยอัลกอริธึม KNN นั้น ได้ใช้ SelectKBest เพื่อหาจำนวนฟีเจอร์ที่มีค่า chi-square มากที่สุดมาใช้ในการสร้างแบบจำลอง โดยพบว่าที่จำนวนฟีเจอร์เท่ากับ 5 ให้ค่า accuracy สูงที่สุด ดังภาพประกอบ 30 ดังนั้นจึงเลือกฟีเจอร์ที่มี chi-square สูงสุด 5 ลำดับ ได้แก่ F7_E23, E15, E23, F7 และ P8 เมื่อปรับพารามิเตอร์ที่ให้ accuracy สูงสุด คือ algorithm: brute, n_neighbors: 15, p: 1, weights: distance ให้ค่าประสิทธิภาพ ดังนี้ Accuracy 0.6494, F1 score 0.6226, Recall 0.6264 และ Precision 0.6553



ภาพประกอบ 30 แสดงจำนวนฟีเจอร์และค่า accuracy เมื่อใช้ฟีเจอร์ที่เรียงลำดับด้วย Cih-square ในการสร้างแบบจำลองด้วยอัลกอริธึม K-Nearest Neighbors

4.4.3 แบบจำลองที่สร้างด้วยอัลกอริธึม Support Vector Machine (SVM)

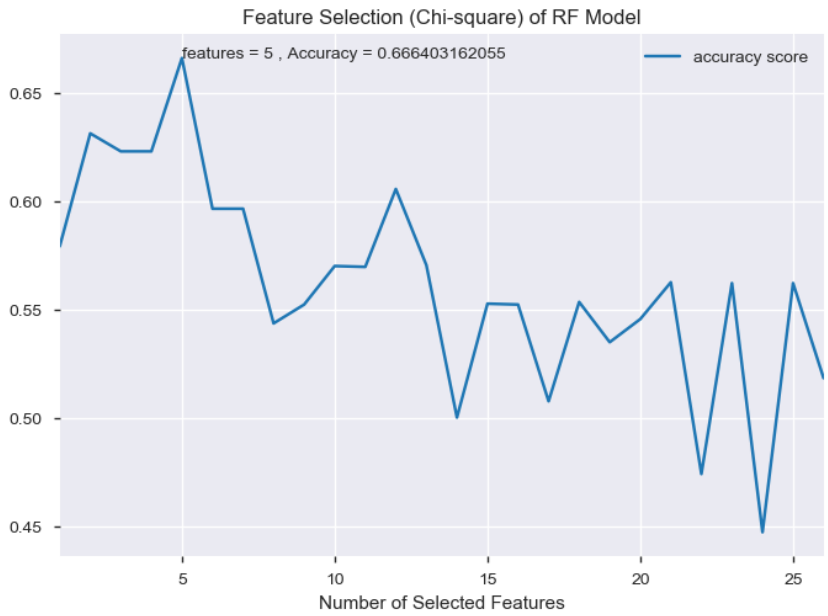
การสร้างแบบจำลองด้วยอัลกอริธึม SVM นั้น ได้ใช้ SelectKBest เพื่อหาจำนวนฟีเจอร์ที่มีค่า chi-square มากที่สุดมาใช้ในการสร้างแบบจำลอง โดยพบว่าที่จำนวนฟีเจอร์เท่ากับ 5 ให้ค่า accuracy สูงที่สุด ดังภาพประกอบ 31 ดังนั้นจึงเลือกฟีเจอร์ที่มี chi-square สูงสุด 3 ลำดับ ได้แก่ F7_E23, E15 และ E23 เมื่อปรับพารามิเตอร์ที่ให้ค่า accuracy สูงสุดคือ C: 4, degree: 1, gamma: 0.1, kernel: 'poly', decision_function_shape: 'ovr' และให้ค่าประสิทธิภาพ ดังนี้ Accuracy 0.6403, F1 score 0.6263, Recall 0.6273 และ Precision 0.6406



ภาพประกอบ 31 แสดงจำนวนฟีเจอร์และค่า accuracy เมื่อใช้ฟีเจอร์ที่เรียงลำดับด้วย Chi-Square ในการสร้างแบบจำลองด้วยอัลกอริธึม Support Vector Machine

4.4.4 แบบจำลองที่สร้างด้วยอัลกอริธึม Random Forest

การสร้างแบบจำลองด้วยอัลกอริธึม Random Forest นั้น ได้ใช้ SelectKBest เพื่อหาจำนวนฟีเจอร์ที่มีค่า chi-square มากที่สุดมาใช้ในการสร้างแบบจำลอง โดยพบว่าที่จำนวนฟีเจอร์เท่ากับ 5 ให้ค่า accuracy สูงที่สุด ดังภาพประกอบ 32 ดังนั้นจึงเลือกฟีเจอร์ที่มี chi-square สูงสุด 5 ลำดับ ได้แก่ F7_E23, E15, E23, F7 และ P8 และปรับพารามิเตอร์ที่ให้ค่า accuracy สูงสุด คือ max_depth: 25, max_features: ‘sqrt’, min_samples_leaf: 3, min_samples_split: 9, n_estimators: 38 และให้ค่าประสิทธิภาพแบบจำลอง ดังนี้ Accuracy 0.6312, F1 score 0.6263, Recall 0.6110 และ Precision 0.6281



ภาพประกอบ 32 แสดงจำนวนฟีเจอร์และค่า accuracy เมื่อใช้ฟีเจอร์ที่เรียงลำดับด้วย Chi-Square ในการสร้างแบบจำลองด้วยอัลกอริธึม Random Forest

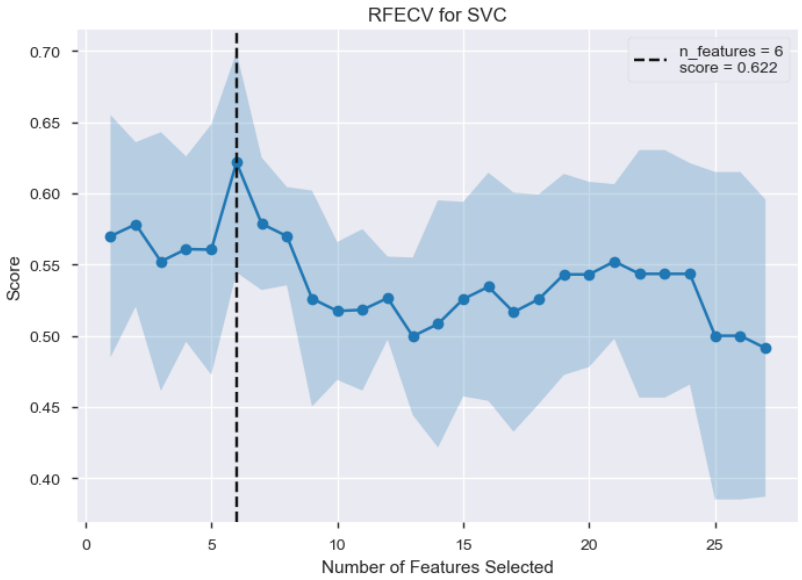
4.5 ผลลัพธ์ของการสร้างแบบจำลองโดยใช้ฟีเจอร์ที่ได้จากการคัดเลือกด้วยวิธี Recursive Feature Elimination (RFE)

การคัดเลือกฟีเจอร์โดยใช้ Recursive Feature Elimination ได้กำหนดวิธีในการวัดความสำคัญของฟีเจอร์ (Feature Importance) โดยแบ่งเป็น 2 แบบ คือ ในแบบจำลองที่สร้างด้วยอัลกอริธึม Random Forest จะใช้ฟีเจอร์ที่มาจากการคัดเลือกด้วย RFE ที่เรียงลำดับความสำคัญของฟีเจอร์โดยวัดด้วยค่า Feature Importance ซึ่งได้จากการนำฟีเจอร์ไปใช้ในแบบจำลอง Random Forest Classifier และในแบบจำลองที่สร้างด้วยอัลกอริธึม Support Vector Machine จะใช้ฟีเจอร์ที่มาจากการคัดเลือกด้วย RFE ที่วัดด้วยค่า Coefficient ส่วนแบบจำลอง Naïve Bayes และ K Nearest Neighbor ไม่สามารถใช้ Feature Importance และ Coefficient ในการคัดเลือกฟีเจอร์ด้วย RFE ได้

4.5.1 แบบจำลองที่สร้างด้วยอัลกอริธึม Support Vector Machine

เมื่อใช้ RFE ซึ่งใช้ Coefficient วัดความสำคัญของฟีเจอร์ ในการคัดเลือกฟีเจอร์ในแบบจำลองที่สร้างด้วยอัลกอริธึม SVM พบว่าจำนวนฟีเจอร์ที่ให้ค่า accuracy สูงที่สุดคือ 6 ตามภาพประกอบ 33 ซึ่งประกอบด้วยฟีเจอร์ F7_E23, E2, E21, F19, P17, P25 และปรับพารามิเตอร์ของแบบจำลอง

ได้ kernel: 'linear', C: 1, degree: 1, gamma: 0.1 ซึ่งให้ประสิทธิภาพของแบบจำลอง ดังนี้ Accuracy 0.6569, F1 score 0.6341, Recall 0.6381 และ Precision 0.6532



ภาพประกอบ 33 แสดงจำนวนฟีเจอร์และค่า accuracy เมื่อใช้ฟีเจอร์ที่คัดเลือกด้วย RFE ที่ลำดับความสำคัญของฟีเจอร์จาก Coefficiency ในการสร้างแบบจำลองด้วยอัลกอริธึม Support Vector Machine

4.5.2 แบบจำลองที่สร้างด้วยอัลกอริธึม Random Forest

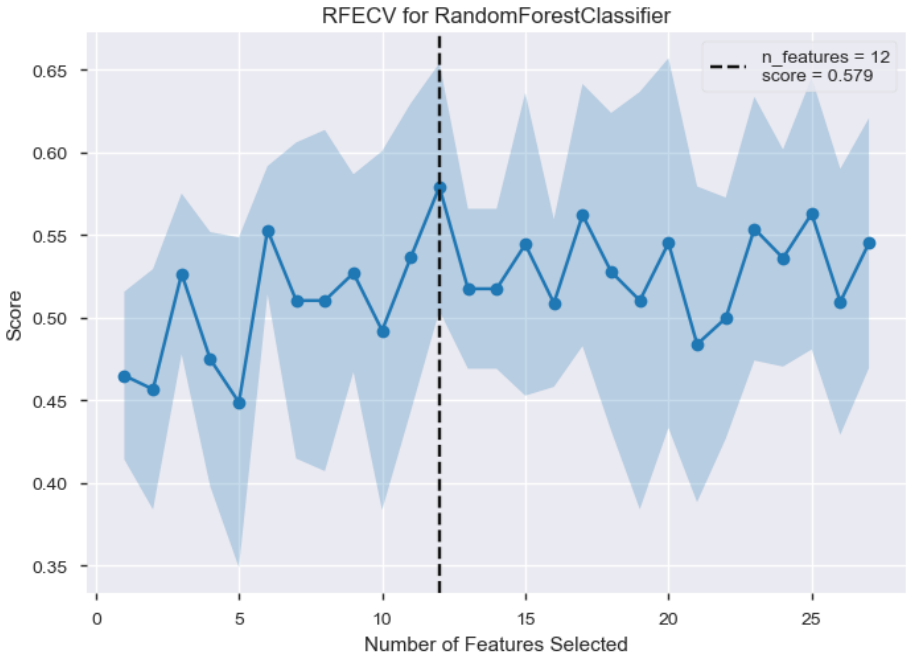
เมื่อใช้ RFE ซึ่งใช้ Feature Importance ในการกำหนดลำดับความสำคัญของฟีเจอร์ ในการคัดเลือกฟีเจอร์ในแบบจำลองที่สร้างด้วยอัลกอริธึม Random Forest พบว่าจำนวนฟีเจอร์ที่ให้ค่า accuracy สูงที่สุดคือ 12 ตามภาพประกอบ 34 ซึ่งประกอบด้วยฟีเจอร์ age, F24, E20, F19, F14, P25, F12, P11, P13, F7_E23, F3 และ P8 โดยปรับพารามิเตอร์ได้ max_depth: 42, min_samples_leaf: 5, min_samples_split: 11, n_estimators: 18 ซึ่งให้ประสิทธิภาพแบบจำลอง ดังนี้ Accuracy 0.6229, F1 score 0.5532, Recall 0.5987 และ Precision 0.6197

ตาราง 4 ลำดับความสำคัญของฟีเจอร์ที่วัดโดยใช้ Feature Importance ด้วยอัลกอริธึม Random Forest

Features	age	F24	E20	F19	F14	P25	F12	P11	P13	F7_E23	F3	P8	F7
Ranking	1	2	3	4	5	6	7	8	9	10	11	12	13

ตารางที่ 4 (ต่อ)

Features	E9	P1	E15	P4	P17	E2	F5	E16	E18	E21	E23	F6	E10	E22
Ranking	14	15	16	17	18	19	20	21	22	23	24	25	26	27

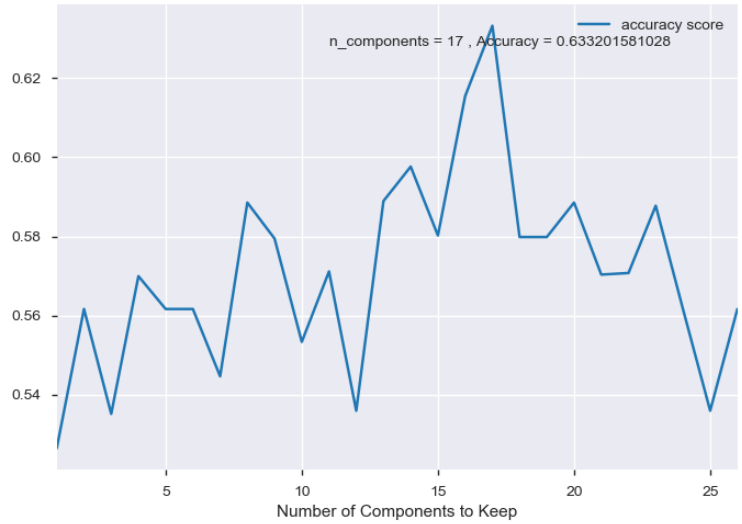


ภาพประกอบ 34 แสดงจำนวนฟีเจอร์และค่า accuracy เมื่อใช้ฟีเจอร์ที่คัดเลือกด้วย RFE ที่ลำดับความสำคัญของฟีเจอร์จาก Feature Importance ในการสร้างแบบจำลองด้วยอัลกอริธึม Random Forest

4.6 ผลลัพธ์ของการสร้างแบบจำลองโดยใช้ฟีเจอร์ที่ได้จากการทำ Principal Component Analysis (PCA)

4.6.1 แบบจำลองที่สร้างด้วยอัลกอริธึม Naïve Bayes

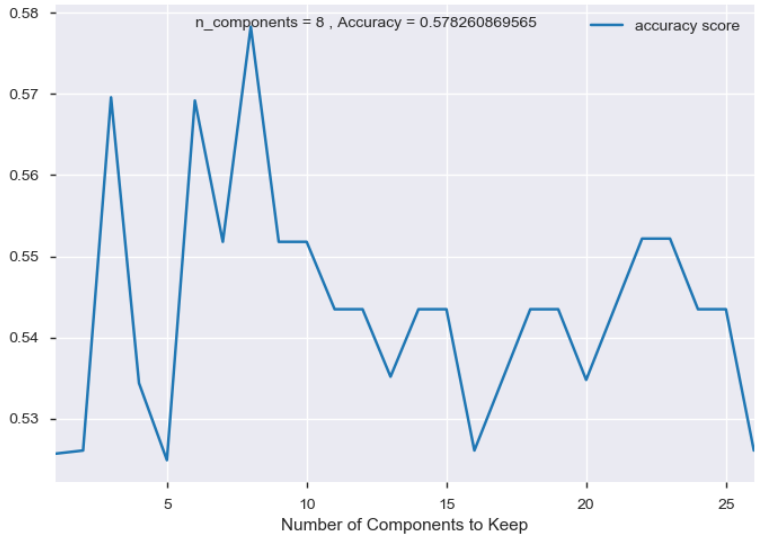
เมื่อหาองค์ประกอบที่ให้ Accuracy ดีที่สุดในการปรับมิติฟีเจอร์ด้วย PCA พบว่าได้องค์ประกอบฟีเจอร์เท่ากับ 17 ตามภาพประกอบ 35 โดยแบบจำลองจะมีประสิทธิภาพดังนี้ Accuracy 0.6332, F1 score 0.6197, Recall 0.6209 และ Precision 0.6282



ภาพประกอบ 35 จำนวนส่วนประกอบหลังการคัดแยกคุณลักษณะเด่นฟีเจอร์ด้วย PCA และค่า accuracy ในการสร้างแบบจำลองด้วยอัลกอริธึม Naïve Bayes

4.6.2 แบบจำลองที่สร้างด้วยอัลกอริธึม K Nearest Neighbors

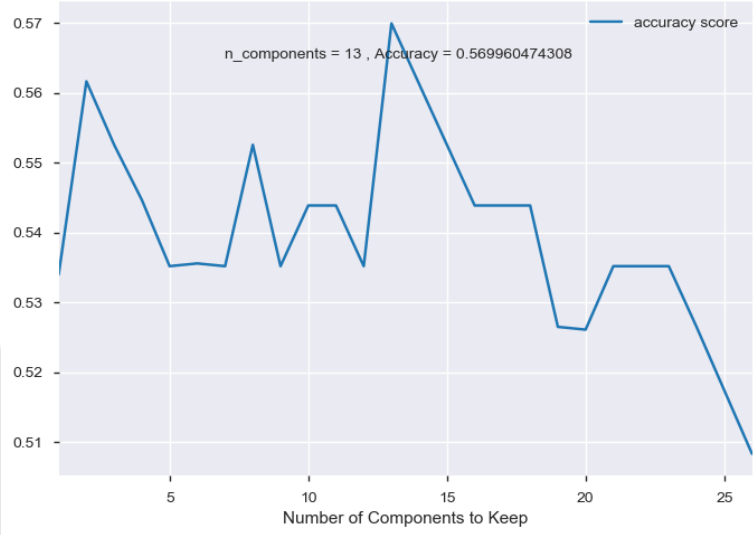
เมื่อหาองค์ประกอบที่ให้ Accuracy ดีที่สุดในการปรับมิติฟีเจอร์ด้วย PCA พบว่าได้องค์ประกอบฟีเจอร์เท่ากับ 8 ตามภาพประกอบ 36 และปรับพารามิเตอร์ได้ algorithm: brute, n_neighbors: 22, p: 1, weights: uniform แบบจำลองจะมีประสิทธิภาพดังนี้ Accuracy score 0.6142, F1 score 0.5663, Recall score 0.5841 และ Precision score 0.6124



ภาพประกอบ 36 จำนวนส่วนประกอบหลังการคัดแยกคุณลักษณะเด่นฟีเจอร์ด้วย PCA และค่า accuracy ในการสร้างแบบจำลองด้วยอัลกอริธึม KNN

4.6.3 แบบจำลองที่สร้างด้วยอัลกอริธึม Support Vector Machine

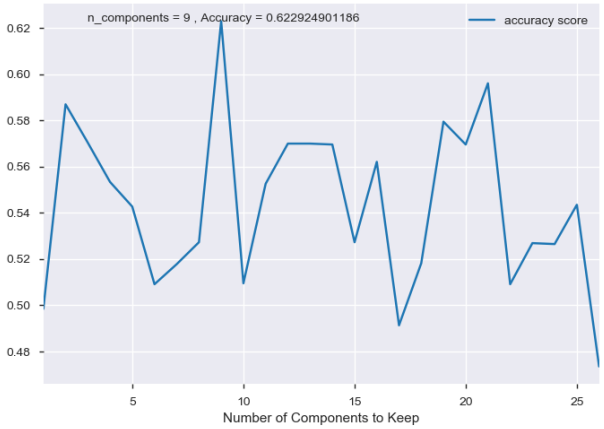
เมื่อหาองค์ประกอบที่ให้ Accuracy ดีที่สุดในการปรับมิติฟีเจอร์ด้วย PCA พบว่าได้องค์ประกอบฟีเจอร์เท่ากับ 13 ตามภาพประกอบ 37 และปรับพารามิเตอร์ได้ C: 1, degree: 1, gamma: 0.1, kernel: poly, decision_function_shape: ovr แบบจำลองจะมีประสิทธิภาพดังนี้ Accuracy: 0.5708, F1 score 0.5482 Recall 0.5525 และ Precision 0.5612



ภาพประกอบ 37 จำนวนส่วนประกอบหลังการคัดแยกคุณลักษณะเด่นฟีเจอร์ด้วย PCA และค่า accuracy ในการสร้างแบบจำลองด้วยอัลกอริธึม SVM

4.6.4 แบบจำลองที่สร้างด้วยอัลกอริธึม Random Forest

เมื่อหาองค์ประกอบที่ให้ Accuracy ดีที่สุดในการปรับมิติฟีเจอร์ด้วย PCA พบว่าได้องค์ประกอบฟีเจอร์เท่ากับ 9 ตามภาพประกอบ 38 และปรับพารามิเตอร์ได้ max_depth: 31, max_features: auto, min_samples_leaf: 14, min_samples_split: 9, n_estimators: 38 แบบจำลองจะมีประสิทธิภาพดังนี้ Accuracy 0.6308, F1 score 0.5798, Recall 0.6018 และ Precision 0.6243



ภาพประกอบ 38 จำนวนส่วนประกอบหลังการคัดแยกคุณลักษณะเด่นฟีเจอร์ด้วย PCA และค่า accuracy ในการสร้างแบบจำลองด้วยอัลกอริธึม Random Forest

บทที่ 5
สรุปผลการวิจัย อภิปรายผล และข้อเสนอแนะ

ในการวิจัยเรื่องการจำแนกประเภทโรคเวียนศีรษะขณะเปลี่ยนท่าจากข้อมูลแบบประเมินอาการวิงเวียนศีรษะโดยใช้เทคนิคการเรียนรู้ของเครื่อง ผู้วิจัยได้วัดประสิทธิภาพของแบบจำลองแต่ละอัลกอริธึม เพื่อนำมาเปรียบเทียบและสรุปผล โดยสามารถแบ่งหัวข้อในการสรุปผลได้ดังต่อไปนี้

1. สรุปผลการวิจัย
2. อภิปรายผลการวิจัย
3. ข้อเสนอแนะ

5.1 สรุปผลการวิจัย

โรคเวียนศีรษะขณะเปลี่ยนท่าเป็นโรคที่ส่งผลกระทบต่อการดำเนินชีวิตของผู้ป่วยอย่างยิ่ง โรคเวียนศีรษะขณะเปลี่ยนท่าที่พบบ่อยมี 2 ประเภท และสามารถวินิจฉัยจำแนกประเภทได้โดยแพทย์ผู้เชี่ยวชาญเท่านั้น อีกทั้งวิธีการกายภาพบำบัดเพื่อรักษาโรคเวียนศีรษะขณะเปลี่ยนท่าที่ต่างประเภทกันก็มีความต่างกัน ดังนั้นการจำแนกประเภทของโรคเวียนศีรษะขณะเปลี่ยนท่าที่ถูกต้องและรวดเร็วจะสามารถช่วยผู้ป่วยและแพทย์ในการดำเนินการรักษาเพื่อยับยั้งอาการที่ส่งผลกระทบต่อผู้ป่วยได้

ในการวิจัยนี้ ผู้วิจัยได้นำข้อมูลจากแบบประเมินอาการวิงเวียนศีรษะมาใช้สร้างแบบจำลองด้วยอัลกอริธึมต่าง ๆ และการคัดเลือกฟีเจอร์ด้วยวิธีที่ต่างกัน โดยใช้เทคนิคการเรียนรู้ของเครื่อง เพื่อเป็นแนวทางในการพัฒนาการสร้างแบบจำลองในการจำแนกประเภทของโรคเวียนศีรษะต่อไป โดยวัดประสิทธิภาพของแต่ละแบบจำลองได้ตามตาราง 5

ตาราง 5 แสดงประสิทธิภาพแบบจำลองที่สร้างจากชุดของฟีเจอร์ที่ต่างกัน และอัลกอริธึมที่ต่างกัน

วิธีการจัดการฟีเจอร์	อัลกอริธึม	ฟีเจอร์ (จำนวนฟีเจอร์)	Accuracy score (%)	Recall score (%)	F1 score (%)	Precision score (%)
-	GNB	age, P1, E2, F3, P4, F5, F6, F7, P8, E9, E10, P11, F12, P13, F14, E15, F16, P17, E18, F19, E20, E21, E22, E23, F24, P25 (26)	54.27	53.61	53.35	53.41
	KNN		52.61	51.18	49.69	50.44
	SVM		57.04	56.12	55.32	56.59
	RF		57.87	54.18	55.31	60.04
คัดเลือกฟีเจอร์โดย Filter Method - chi-square	GNB	F7_E23, E15, E23, F7, P8, F24 (6)	67.51	65.97	65.95	67.20
	KNN	F7_E23, E15, E23, F7, P8 (5)	64.94	62.64	62.26	65.53
	SVM	F7_E23, E15, E23 (3)	64.03	62.73	62.63	64.06
	RF	F7_E23, E15, E23, F7, P8 (5)	63.12	61.10	62.63	62.81
คัดเลือกฟีเจอร์โดย Wrapper Method - RFE	GNB	-	-	-	-	-
	KNN	-	-	-	-	-
	SVM	F7_E23, E2, E21, F19, P17, P25 (6)	65.69	63.81	63.41	65.32
	RF	age, F24, E20, F19, F14, P25, F12, P11, P13, F7_E23, F3, P8 (12)	62.29	59.87	55.32	61.97
Feature Extraction-PCA	GNB	n_component = 17	63.32	62.09	61.97	62.82
	KNN	n_component = 8	61.42	58.41	56.63	61.24
	SVM	n_component = 13	57.08	55.25	54.82	56.12
	RF	n_component = 9	63.08	60.18	57.98	62.43

แบบจำลองที่มีประสิทธิภาพที่สุดได้จากการสร้างด้วยอัลกอริธึม Gaussian Naïve Bayes โดยใช้ฟีเจอร์ที่ผ่านการคัดเลือกด้วย chi-square ได้แก่ F7_E23, E15, E23, F7, P8 และ F24 โดยให้ค่าประสิทธิภาพ ดังนี้ Accuracy score 67.51%, Recall score 65.97%, F1 score 65.95% และ Precision score 67.20%

5.2 อภิปรายผลการวิจัย

จากตาราง 5 แบบจำลองสร้างด้วยอัลกอริธึม Gaussian Naïve Bayes ให้ประสิทธิภาพดีที่สุด เนื่องจากเป็นอัลกอริธึมที่เหมาะกับข้อมูลขนาดเล็ก แต่ประสิทธิภาพที่ได้ยังไม่เพียงพอสำหรับการใช้ในทางการแพทย์ จึงต้องมีการพัฒนาแบบจำลองให้มีประสิทธิภาพสูงขึ้นเพื่อนำไปใช้จริงได้

5.3 ข้อเสนอแนะ

1. เนื่องจากโรคเวียนศีรษะขณะเปลี่ยนท่าเป็นโรคที่พบได้ไม่บ่อย ดังนั้นหากมีการเก็บข้อมูลได้มากขึ้นกว่านี้ อาจจะเหมาะสมกับแบบจำลองที่สร้างด้วยอัลกอริธึม Random Forest มากกว่า
2. ในการวิจัยครั้งนี้เมื่อคัดเลือกฟีเจอร์ด้วย chi-square และ RFE พบว่าฟีเจอร์ที่ได้มีฟีเจอร์ที่สร้างใหม่ คือ F7_E23 ทั้งสิ้น ฟีเจอร์ F7_E23 จึงน่าจะเป็นฟีเจอร์สำคัญที่ช่วยในการพัฒนาการสร้างแบบจำลองเพื่อจำแนกประเภทโรคเวียนศีรษะขณะเปลี่ยนท่าต่อไปได้
3. การหาฟีเจอร์เพิ่มเติม เช่น ระยะเวลาที่เกิดอาการเวียนศีรษะ และ ระยะเวลาการดำเนินโรค อาจจะเป็นฟีเจอร์สำคัญที่ทำให้แบบจำลองจำแนกประเภทเวียนศีรษะขณะเปลี่ยนท่ามีประสิทธิภาพที่สูงขึ้น

ภาคผนวก

รายละเอียดโค้ดที่ใช้ในการวิจัย

```
#import package ที่จำเป็น
!pip install pandas-profiling
import pandas_profiling
from sklearn.ensemble import ExtraTreesClassifier
import os
import pandas as pd
import numpy as np
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import zscore
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_validate
from sklearn.model_selection import ShuffleSplit
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.decomposition import PCA
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.feature_selection import RFE
from sklearn.datasets import make_classification
!pip install yellowbrick
from yellowbrick.features import RFECV
from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
#นำเข้าไฟล์ข้อมูล
os.chdir('C:\\Users\\User\\Desktop')
file_name = 'bppv dataset.xlsx'
df = pd.read_excel(file_name)
df
#ตัดคอลัมน์ที่มี null เยอะ
df.isnull().sum()).sort_values(ascending=False)
df.drop("nyst",axis=1,inplace=True)
#ดูรายละเอียดข้อมูล
pandas_profiling.ProfileReport(df)

df.describe()

#สร้าง Age distribution
df[df['diagnosis']=='pcb']['age'].mode()
fig, axes = plt.subplots(1, 2)
data1 = df[df['diagnosis']=='pcb']['age']
data2 = df[df['diagnosis']=='hcb']['age']
plt.title('Age distribution ')
axes[0].hist(x=data1,color='red')
axes[1].hist(x=data2)
fig.tight_layout()
#สร้าง age distribution โดยแบ่งตามช่วงอายุ
def combine_age(age):
    if age<30:
        return '30'
    if age == 30 or age>30 and age<40:
        return '30-40'
    if age == 40 or age>40 and age<50:
        return '40-50'
    if age == 50 or age>50 and age<60:
        return '50-60'
    if age == 60 or age>60 and age<70:
```

```
            return '60-70'
        if age == 70 or age>70 and age<80:
            return '70-80'
        if age == 80 or age>80:
            return '80+'
        else:
             return age

df_age_distribution = df.copy()
df_age_distribution['age']=df_age_distribution['age'].apply(combine_age)
df_age = df_age_distribution['age'].value_counts()
df_age.values
df_age_pcb = df_age_distribution[df_age_distribution['diagnosis']=='pcb']['age'].value_counts()
df_age_hcb = df_age_distribution[df_age_distribution['diagnosis']=='hcb']['age'].value_counts()
fig, axes = plt.subplots(1, 2,figsize=(8,5))
age1 = df_age_hcb.index.tolist()
count1 = df_age_hcb.values.tolist()
age2 = df_age_pcb.index.tolist()
count2 = df_age_pcb.values.tolist()
axes[0].barh(age1, count1, height=0.5, align='center', color='crimson')
axes[0].set_xlabel('frequency (hcb)')
axes[0].set_ylabel('age')
axes[1].barh(age2, count2, height=0.5, align='center', color='darkblue')
axes[1].set_xlabel('frequency (pcb)');

sns.distplot(df[df['diagnosis']=='pcb']['age'] , color="skyblue", label="pcb")
sns.distplot(df[df['diagnosis']=='hcb']['age'] , color="red", label="hcb")
sns.despine(offset=20, trim=False)
#เปลี่ยนรูปแบบของ diagnosis ในข้อมูลจากข้อความให้เป็นตัวเลข
df['diagnosis'].replace(['hcb','pcb'],[1,0],inplace=True)
#จัดเรียงฟีเจอร์
df = df[['no', 'onset', 'age', 'P1', 'E2', 'F3', 'P4', 'F5', 'F6', 'F7', 'P8',
       'E9', 'E10', 'P11', 'F12', 'P13', 'F14', 'E15', 'F16', 'P17', 'E18',
       'F19', 'E20', 'E21', 'E22', 'E23', 'F24', 'P25',  'diagnosis']]
#ระบุ label และข้อมูลสำหรับฝึก
Y=target = df.loc[:, 'diagnosis']
X=data = df[['age', 'P1', 'E2', 'F3', 'P4', 'F5', 'F6', 'F7', 'P8',
       'E9', 'E10', 'P11', 'F12', 'P13', 'F14', 'E15', 'F16', 'P17', 'E18',
       'F19', 'E20', 'E21', 'E22', 'E23', 'F24', 'P25']]
#ดูความสำคัญของฟีเจอร์
model = ExtraTreesClassifier(random_state=42)
model.fit(X, Y)
model.feature_importances_
indexes = ['age', 'P1', 'E2', 'F3', 'P4', 'F5', 'F6', 'F7', 'P8',
       'E9', 'E10', 'P11', 'F12', 'P13', 'F14', 'E15', 'F16', 'P17', 'E18',
       'F19', 'E20', 'E21', 'E22', 'E23', 'F24', 'P25']
feature_imp = pd.Series(model.feature_importances_, index=indexes).sort_values(ascending=False)
ax = feature_imp.plot(kind='bar',color='blue')
ax.set(ylabel='Feature Importance')

# สร้างแบบจำลองด้วย GNB โดยใช้ฟีเจอร์ทั้งหมด
gnb_clf = GaussianNB()
cv = 5 #กำหนดการแบ่งข้อมูลสำหรับทำ Cross Validation เพื่อทดสอบแบบจำลองเป็น 5 ส่วน
f1score = cross_val_score(gnb_clf,data,target, cv=cv,scoring='f1_macro')
#วัดประสิทธิภาพของแบบจำลองโดยใช้ f1-score
accscore = cross_val_score(gnb_clf,data,target, cv=cv) #วัดประสิทธิภาพของแบบจำลองโดยใช้ accuracy score
recallscore = cross_val_score(gnb_clf,data,target, cv=cv,scoring='recall_macro')
#วัดประสิทธิภาพของแบบจำลองโดยใช้ recall score
pscore = cross_val_score(gnb_clf,data,target, cv=cv,scoring='precision_macro')
#วัดประสิทธิภาพของแบบจำลองโดยใช้ precision score
f_gnb = np.mean(f1score) #หาค่าเฉลี่ยของค่าประสิทธิภาพ f1-score ที่ได้ทั้งหมด 5 ครั้ง
acc_gnb = np.mean(accscore) #หาค่าเฉลี่ยของค่าประสิทธิภาพ accuracy score ที่ได้ทั้งหมด 5 ครั้ง
r_gnb = np.mean(recallscore) #หาค่าเฉลี่ยของค่าประสิทธิภาพ recall score ที่ได้ทั้งหมด 5 ครั้ง
p_gnb = np.mean(pscore) #หาค่าเฉลี่ยของค่าประสิทธิภาพ precision score ที่ได้ทั้งหมด 5 ครั้ง
print ('Accuracy:', acc_gnb)
print ('F1 score:', f_gnb)
print ('Recall:', r_gnb)
print ('Precision:', p_gnb)
```

```
138
139 # สร้างแบบจำลองด้วย KNN โดยใช้ฟีเจอร์ทั้งหมด
140 knn = KNeighborsClassifier()
141 #ปรับพารามิเตอร์
142 k_range = list(range(1, 31))
143 p = [1,2]
144 weights=['distance','uniform']
145 param_grid = dict(n_neighbors=k_range,p=p,weights=weights)
146 grid = GridSearchCV(knn, param_grid, cv=5, scoring='accuracy')
147 grid.fit(X,Y)
148 print(grid.best_params_)
149 print(grid.best_estimator_)
150 #วัดประสิทธิภาพของแบบจำลอง
151 clf_KNNd = KNeighborsClassifier(n_neighbors=2,p=1,weights='distance')
152 f1score = cross_val_score(clf_KNNd,data,target, cv=cv,scoring='f1_macro')
153 accscore = cross_val_score(clf_KNNd,data,target, cv=cv)
154 recallscore = cross_val_score(clf_KNNd,data,target, cv=cv,scoring='recall_macro')
155 pscore = cross_val_score(clf_KNNd,data,target, cv=cv,scoring='precision_macro')
156 f_knn = np.mean(f1score)
157 acc_knn = np.mean(accscore)
158 r_knn = np.mean(recallscore)
159 p_knn = np.mean(pscore)
160 print ('Accuracy:', acc_knn)
161 print ('F1 score:', f_knn)
162 print ('Recall:', r_knn)
163 print ('Precision:', p_knn)
164
165 # สร้างแบบจำลองด้วย SVM โดยใช้ฟีเจอร์ทั้งหมด
166 #ปรับพารามิเตอร์
167 C=range(1,10)
168 degree = range(1,3)
169 gamma = (0.1,5)
170 param_grid=dict(C=C,degree=degree,gamma=gamma)
171 grid=GridSearchCV(SVC( kernel='poly',decision_function_shape='ovr',
    random_state=42),param_grid,cv=5)
172 grid.fit(X, Y)
173 print(grid.best_params_)
174 print(grid.best_estimator_)
175 #วัดประสิทธิภาพของแบบจำลอง
176 clf_SVC = SVC( C=1,degree=2,gamma=0.1,kernel='poly',decision_function_shape='ovr',
    random_state=42)
177 f1score = cross_val_score(clf_SVC,data,target, cv=cv,scoring='f1_macro')
178 accscore = cross_val_score(clf_SVC,data,target, cv=cv)
179 recallscore = cross_val_score(clf_SVC,data,target, cv=cv,scoring='recall_macro')
180 pscore = cross_val_score(clf_SVC,data,target, cv=cv,scoring='precision_macro')
181 f_svc = np.mean(f1score)
182 acc_svc = np.mean(accscore)
183 r_svc = np.mean(recallscore)
184 p_svc = np.mean(pscore)
185 print ('Accuracy:', acc_svc)
186 print ('F1 score:', f_svc)
187 print ('Recall:', r_svc)
188 print ('Precision:', p_svc)
189
190 # สร้างแบบจำลองด้วย Random Forestโดยใช้ฟีเจอร์ทั้งหมด
191 n_estimators = list(range(2, 50))
192 max_features = ['auto', 'sqrt']
193 max_depth = list(range(2, 50))
194 min_samples_split = list(range(2, 20))
195 min_samples_leaf = list(range(2, 20))
196 random_grid = dict(n_estimators= n_estimators,
197                    max_features = max_features,
198                    max_depth= max_depth,
199                    min_samples_split= min_samples_split,
200                    min_samples_leaf= min_samples_leaf)
201 rf = RandomForestClassifier(random_state=42)
202 rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,
    n_iter = 50, cv = 5,
203                                verbose=2, random_state=42, n_jobs = -1)
204 rf_random.fit(X, Y)
205 rf_random.best_params_
206 #วัดประสิทธิภาพของแบบจำลอง
```

```
clf_RFC =  RandomForestClassifier(max_depth=6,max_features='auto',min_samples_leaf=10,
                                  min_samples_split=14,n_estimators=30,random_state=42
                                  )
accscore = cross_val_score(clf_RFC,data,target, cv=cv)
recallscore = cross_val_score(clf_RFC,data,target, cv=cv,scoring='recall_macro')
pscore = cross_val_score(clf_RFC,data,target, cv=cv,scoring='precision_macro')
F1_RFC = np.mean(f1score)
acc_RFC = np.mean(accscore)
recall_RFC = np.mean(recallscore)
p_RFC = np.mean(pscore)
print ('Accuracy:', acc_RFC)
print ('F1 score:', F1_RFC)
print ('Recall:', recall_RFC)
print ('Precision:', p_RFC)

# หาค่าสัมประสิทธิ์สหสัมพันธ์แบบเพียร์สันของแต่ละฟีเจอร์
df2 = df[['age', 'P1', 'E2', 'F3', 'P4', 'F5', 'F6', 'F7', 'P8',
       'E9', 'E10', 'P11', 'F12', 'P13', 'F14', 'E15', 'F16', 'P17', 'E18',
       'F19', 'E20', 'E21', 'E22', 'E23', 'F24', 'P25',
       'diagnosis']]#เรียงลำดับฟีเจอร์ตามแบบประเมินอาการวิงเวียนศีรษะ
fig, ax = plt.subplots(figsize=(20,20))
sns.heatmap(df2.corr(method='pearson'), annot=True, fmt=".2f",linewidths=.5, ax=ax)
#สร้างกราฟแสดงค่าสัมประสิทธิ์สหสัมพันธ์แบบเพียร์สัน
#สร้างฟีเจอร์ F7_E23
F7_E23 = df['F7']+df['E23']
df['F7_E23'] = pd.Series(F7_E23)
# หาค่าสัมประสิทธิ์สหสัมพันธ์แบบเพียร์สันของฟีเจอร์ใหม่เทียบกับฟีเจอร์เก่า
df3 = df[['F7','E23', 'F7_E23','diagnosis']]
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(df3.corr(method='pearson'), annot=True, fmt=".2f",linewidths=.5, ax=ax)

# The Recursive Feature Elimination (or RFE)
# RFE with RF
y=target = df.loc[:, 'diagnosis']
X=data = df[['age', 'P1', 'E2', 'F3', 'P4', 'F5', 'F6', 'F7', 'P8',
       'E9', 'E10', 'P11', 'F12', 'P13', 'F14', 'E15', 'F16', 'P17', 'E18',
       'F19', 'E20', 'E21', 'E22', 'E23', 'F24', 'P25', 'F7_E23']]
cv = StratifiedKFold(5)
rfe = RFECV(RandomForestClassifier(random_state=42), cv=cv, scoring='accuracy')
rfe.fit(X,y)
rfe.poof()
#เรียงลำดับความสำคัญของฟีเจอร์
ranking_ds=pd.DataFrame({"Features":X.columns})
fit=rfe.fit(X,y)
ref_ranking=rfe.ranking_
ranking_ds['Ranking']=ref_ranking
ranking_ds = ranking_ds.sort_values(by='Ranking', ascending=1)
ranking_ds
ranking_ds['Features'].tolist()
#สร้างแบบจำลอง และปรับพารามิเตอร์
Xnew = df[['age','F24','E20','F19','F14','P25','F12','P11','P13','F7_E23','F3','P8',]]
n_estimators = list(range(2, 50))
max_features = ['auto', 'sqrt']
max_depth = list(range(2, 50))
min_samples_split = list(range(2, 20))
min_samples_leaf = list(range(2, 20))
print(n_estimators,max_features,max_depth,min_samples_split,min_samples_leaf)
random_grid = dict(n_estimators= n_estimators,
                   max_features = max_features,
                   max_depth= max_depth,
                   min_samples_split= min_samples_split,
                   min_samples_leaf= min_samples_leaf)
rf = RandomForestClassifier(random_state=42)
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,
n_iter = 50, cv = 5,
                               verbose=2, random_state=42, n_jobs = -1)
rf_random.fit(Xnew, Y)
rf_random.best_params_
#วัดประสิทธิภาพของแบบจำลอง
clf_RFC =
RandomForestClassifier(max_depth=42,max_features='sqrt',min_samples_leaf=5,min_samples
```

```
    _split=11,
                                        n_estimators=18,random_state=42)
accscore = cross_val_score(clf_RFC,Xnew,target, cv=cv)
recallscore = cross_val_score(clf_RFC,Xnew,target, cv=cv,scoring='recall_macro')
pscore = cross_val_score(clf_RFC,Xnew,target, cv=cv,scoring='precision_macro')
F1_RFC = np.mean(f1score)
acc_RFC = np.mean(accscore)
recall_RFC = np.mean(recallscore)
p_RFC = np.mean(pscore)
print ('Accuracy:', acc_RFC)
print ('F1 score:', F1_RFC)
print ('Recall:', recall_RFC)
print ('Precision:', p_RFC)

# RFE with SVM
viz = RFECV(SVC(kernel='linear', random_state = 42), cv=5, scoring='accuracy')
viz.fit(data,target)
viz.poof()
#เรียงลำดับความสำคัญของฟีเจอร์
ranking_ds=pd.DataFrame({"Features":X.columns})
fit=viz.fit(data,target)
ref_ranking=viz.ranking_
ranking_ds['Ranking']=ref_ranking
ranking_ds = ranking_ds.sort_values(by='Ranking', ascending=1)
ranking_ds

ranking_ds['Features'].tolist()
#ปรับพารามิเตอร์
y= df.loc[:, 'diagnosis']
Xnew = df[['F7_E23','E2','E21','F19','P17','P25']]
C=range(1,10)
degree = range(1,3)
gamma = (0.1,10)
param_grid={"C":C,"degree":degree,"gamma":gamma}
grid=GridSearchCV(SVC(random_state=42,kernel='poly'),param_grid=param_grid,cv=5)
grid.fit(Xnew,y)
print(grid.best_params_)
print(grid.best_estimator_)
#วัดประสิทธิภาพของแบบจำลอง
clf_SVC = SVC( C=1,degree=1,gamma=0.1,kernel='poly',decision_function_shape='ovr',
random_state=42)
f1score = cross_val_score(clf_SVC,Xnew,target, cv=cv,scoring='f1_macro')
accscore = cross_val_score(clf_SVC,Xnew,target, cv=cv)
recallscore = cross_val_score(clf_SVC,Xnew,target, cv=cv,scoring='recall_macro')
pscore = cross_val_score(clf_SVC,Xnew,target, cv=cv,scoring='precision_macro')
f_svc = np.mean(f1score)
acc_svc = np.mean(accscore)
r_svc = np.mean(recallscore)
p_svc = np.mean(pscore)
print ('Accuracy:', acc_svc)
print ('F1 score:', f_svc)
print ('Recall:', r_svc)
print ('Precision:', p_svc)

# หาค่า Chi-square
y= df.loc[:, 'diagnosis']
X= df[['age', 'P1', 'E2', 'F3', 'P4', 'F5', 'F6', 'F7', 'P8',
       'E9', 'E10', 'P11', 'F12', 'P13', 'F14', 'E15', 'F16', 'P17', 'E18',
       'F19', 'E20', 'E21', 'E22', 'E23', 'F24', 'P25', 'F7_E23']]
test = SelectKBest(score_func=chi2, k=5)

names = ['age', 'P1', 'E2', 'F3', 'P4', 'F5', 'F6', 'F7', 'P8',
       'E9', 'E10', 'P11', 'F12', 'P13', 'F14', 'E15', 'F16', 'P17', 'E18',
       'F19', 'E20', 'E21', 'E22', 'E23', 'F24', 'P25', 'F7_E23']
feature = list()
chi = list()
for n in range(0, len(names)):
    test = SelectKBest(score_func=chi2, k=5)
    fit = test.fit(X,y)
    feature.append(names[n])
    chi.append(fit.scores_[n])

```

```
343         print(names[n], fit.scores_[n])
344     #เรียงลำดับฟีเจอร์ที่มีค่า Chi-square สูงสุดไปต่ำสุด
345     ranking_chi=pd.DataFrame({"ChiSquared": chi,"Features":feature})
346     ranking_chi = ranking_chi.sort_values(by='ChiSquared', ascending=0)
347     ranking_chi = ranking_chi[['Features','ChiSquared']]
348     ranking_chi
349
350
351     #หาจำนวนฟีเจอร์ที่ให้ accuracy ดีที่สุด เมื่อคัดเลือกจาก Chi-square เพื่อสร้างแบบจำลองด้วย GNB
352     accGNB = list()
353     n_features = list()
354
355     for n in range(1,27):
356         Xnew =  SelectKBest(chi2, k=n).fit_transform(X, y)
357         x_train, x_test, y_train, y_test = train_test_split(Xnew, target, test_size=0.2,
        random_state=42)
358         gnb_clf = GaussianNB()
359         gnb_clf.fit(x_train, y_train)
360         gnb_pred = gnb_clf.predict(x_test)
361         accscore = cross_val_score(gnb_clf,Xnew,y, cv=5)
362         acc_gnb = np.mean(accscore)
363         accGNB.append(acc_gnb)
364         n_features.append(n)
365
366     Chi2GNB=pd.DataFrame({"Number of Selected Features":n_features})
367     Chi2GNB['accuracy score']= accGNB #สร้าง DataFrame จำนวนฟีเจอร์และ accuracy ที่ได้
368     Chi2GNB.sort_values(by='Number of Selected Features', ascending=1) #เรียงลำดับตามจำนวนฟีเจอร์
369
370     #สร้างกราฟแสดงความสัมพันธ์ระหว่างจำนวนฟีเจอร์และ accuracy และหาจำนวนฟีเจอร์ที่ให้ค่า accuracy สูงสุด
371     NumberofSelectedFeatures = Chi2GNB['Number of Selected Features']
372     accuracy_score = Chi2GNB['accuracy score']
373     ax = plt.gca()
374     plt.title('Feature Selection (Chi-square) of Naive Bayes Model')
375     Chi2GNB.plot(kind='line',x='Number of Selected Features',y='accuracy score',ax=ax)
376     max_index = np.where(accuracy_score == max(accuracy_score))
377     accuracy_score_max = accuracy_score[max_index[0][0]]
378     NumberofSelectedFeatures_max = NumberofSelectedFeatures[max_index[0][0]]
379     maxValue = 'features ='+str(NumberofSelectedFeatures_max)+' , '+'Accuracy score
        ='+str(accuracy_score_max)
380     plt.annotate(maxValue, xy=(NumberofSelectedFeatures_max,accuracy_score_max))
381     plt.show()
382     #วัดประสิทธิภาพของแบบจำลอง
383     y= df.loc[:, 'diagnosis']
384     X= df[['age', 'P1', 'E2', 'F3', 'P4', 'F5', 'F6', 'F7', 'P8',
385            'E9', 'E10', 'P11', 'F12', 'P13', 'F14', 'E15', 'F16', 'P17', 'E18',
386            'F19', 'E20', 'E21', 'E22', 'E23', 'F24', 'P25', 'F7_E23']]
387     Xnew =  SelectKBest(chi2, k=6).fit_transform(X, y)
388     gnb_clf = GaussianNB()
389     gnb_clf.fit(x_train, y_train)
390     gnb_pred = gnb_clf.predict(x_test)
391     cv = 5
392     flscore = cross_val_score(gnb_clf,Xnew,y, cv=cv,scoring='f1_macro')
393     accscore = cross_val_score(gnb_clf,Xnew,y, cv=cv)
394     recallscore = cross_val_score(gnb_clf,Xnew,y, cv=cv,scoring='recall_macro')
395     pscore = cross_val_score(gnb_clf,Xnew,y, cv=cv,scoring='precision_macro')
396     f_gnb = np.mean(flscore)
397     acc_gnb = np.mean(accscore)
398     r_gnb = np.mean(recallscore)
399     p_gnb = np.mean(pscore)
400     print ('Accuracy:', acc_gnb)
401     print ('F1 score:', f_gnb)
402     print ('Recall:', r_gnb)
403     print ('Precision:', p_gnb)
404     #ดูฟีเจอร์ที่ใช้
405     selector = SelectKBest(chi2, k=6).fit(X, y)
406     X.columns[selector.get_support(indices=True)]
407     vector_names = list(X.columns[selector.get_support(indices=True)])
408     print(vector_names)
409     #หาจำนวนฟีเจอร์ที่ให้ accuracy ดีที่สุด เมื่อคัดเลือกจาก Chi-square เพื่อสร้างแบบจำลองด้วย KNN
410     n_features = list()
411     accKNN = list()
412     for n in range(1,27):
```

```
    Xnew =  SelectKBest(chi2, k=n).fit_transform(X, y)
    x_train, x_test, y_train, y_test = train_test_split(Xnew, target, test_size=0.2,
    random_state=42)
    knn_clf = KNeighborsClassifier()
    knn_clf.fit(x_train, y_train)
    knn_pred = knn_clf.predict(x_test)
    accscore = cross_val_score(knn_clf,Xnew,y, cv=5)
    acc_knn = np.mean(accscore)
    accKNN.append(acc_knn)
    n_features.append(n)

Chi2KNN=pd.DataFrame({"Number of Selected Features":n_features})

Chi2KNN['accuracy score']= accKNN
Chi2KNN.sort_values(by='Number of Selected Features', ascending=1)
NumberofSelectedFeatures = Chi2KNN['Number of Selected Features']
accuracy_score = Chi2KNN['accuracy score']
ax = plt.gca()
max_index = np.where(accuracy_score == max(accuracy_score))
accuracy_score_max = accuracy_score[max_index[0][0]]
NumberofSelectedFeatures_max = NumberofSelectedFeatures[max_index[0][0]]
maxValue = 'features = '+str(NumberofSelectedFeatures_max)+' , '+'Accuracy =
'+str(accuracy_score_max)
plt.annotate(maxValue, xy=(NumberofSelectedFeatures_max,accuracy_score_max))
plt.title('Feature Selection (Chi-square) of KNN Model')
Chi2KNN.plot(kind='line',x='Number of Selected Features',y='accuracy score',ax=ax)
plt.show()
#ปรับพารามิเตอร์
knn = KNeighborsClassifier()
Xnew =  SelectKBest(chi2, k=6).fit_transform(X, y)
k_range = list(range(1, 31))
p = [1,2]
weights=['distance','uniform']
param_grid = dict(n_neighbors=k_range,p=p,weights=weights)
grid = GridSearchCV(knn, param_grid, cv=5, scoring='accuracy')
grid.fit(Xnew,y)
print(grid.best_params_)
print(grid.best_estimator_)
#วัดประสิทธิภาพของแบบจำลอง
clf_KNNd = KNeighborsClassifier(n_neighbors=2,p=1,weights='distance')
f1score = cross_val_score(clf_KNNd,Xnew,target, cv=cv,scoring='f1_macro')
accscore = cross_val_score(clf_KNNd,Xnew,target, cv=cv)
recallscore = cross_val_score(clf_KNNd,Xnew,target, cv=cv,scoring='recall_macro')
pscore = cross_val_score(clf_KNNd,Xnew,target, cv=cv,scoring='precision_macro')
f_knn = np.mean(f1score)
acc_knn = np.mean(accscore)
r_knn = np.mean(recallscore)
p_knn = np.mean(pscore)
print ('Accuracy:', acc_knn)
print ('F1 score:', f_knn)
print ('Recall:', r_knn)
print ('Precision:', p_knn)
#หาจำนวนฟีเจอร์ที่ให้ accuracy ดีที่สุด เมื่อคัดเลือกจาก Chi-square เพื่อสร้างแบบจำลองด้วย SVM
n_features = list()
accSVC = list()
for n in range(1,27):
    Xnew =  SelectKBest(chi2, k=n).fit_transform(X, y)
    svc_clf = SVC( random_state=42)
    accscore = cross_val_score(svc_clf,Xnew,y, cv=5)
    acc_svc = np.mean(accscore)
    accSVC.append(acc_svc)
    n_features.append(n)

Chi2SVC=pd.DataFrame({"Number of Selected Features":n_features})

Chi2SVC['accuracy score']= accSVC
Chi2SVC.sort_values(by='Number of Selected Features', ascending=1)

NumberofSelectedFeatures = Chi2SVC['Number of Selected Features']
accuracy_score = Chi2SVC['accuracy score']
ax = plt.gca()
max_index = np.where(accuracy_score == max(accuracy_score))
```

```
accuracy_score_max = accuracy_score[max_index[0][0]]
NumberofSelectedFeatures_max = NumberofSelectedFeatures[max_index[0][0]]
maxValue = 'features = '+str(NumberofSelectedFeatures_max)+' , '+'Accuracy = '+str(accuracy_score_max)
plt.annotate(maxValue, xy=(NumberofSelectedFeatures_max,accuracy_score_max))

ax = plt.gca()
plt.title('Feature Selection (Chi-square) of SVM Model')
Chi2SVC.plot(kind='line',x='Number of Selected Features',y='accuracy score',ax=ax)
plt.show()
#ปรับพารามิเตอร์
Xnew =  SelectKBest(chi2, k=5).fit_transform(X, y)
C=range(1,10)
degree = range(1,3)
gamma = (0.1,5)
param_grid=dict(C=C,degree=degree,gamma=gamma)
grid=GridSearchCV(SVC(random_state=42),param_grid,cv=5)
grid.fit(Xnew, Y)

print(grid.best_params_)
print(grid.best_estimator_)
#วัดประสิทธิภาพของแบบจำลอง
clf_SVC = SVC( C=4,degree=1,gamma=0.1, random_state=42)
f1score = cross_val_score(clf_SVC,Xnew,target, cv=cv,scoring='f1_macro')
accscore = cross_val_score(clf_SVC,Xnew,target, cv=cv)
recallscore = cross_val_score(clf_SVC,Xnew,target, cv=cv,scoring='recall_macro')
pscore = cross_val_score(clf_SVC,Xnew,target, cv=cv,scoring='precision_macro')
f_svc = np.mean(f1score)
acc_svc = np.mean(accscore)
r_svc = np.mean(recallscore)
p_svc = np.mean(pscore)
print ('Accuracy:', acc_svc)
print ('F1 score:', f_svc)
print ('Recall:', r_svc)
print ('Precision:', p_svc)

#หาจำนวนฟีเจอร์ที่ให้ accuracy ดีที่สุด เมื่อคัดเลือกจาก Chi-square เพื่อสร้างแบบจำลองด้วย RandomForestClassifier
n_features = list()
accRF = list()
for n in range(1,27):
    Xnew =  SelectKBest(chi2, k=n).fit_transform(X, y)
    rf_clf = RandomForestClassifier(random_state=42)
    accscore = cross_val_score(rf_clf,Xnew,y, cv=5)
    acc_rf = np.mean(accscore)
    accRF.append(acc_rf)
    n_features.append(n)

Chi2RF=pd.DataFrame({"Number of Selected Features":n_features})

Chi2RF['accuracy score']= accRF
Chi2RF.sort_values(by='Number of Selected Features', ascending=1)

NumberofSelectedFeatures = Chi2RF['Number of Selected Features']
accuracy_score = Chi2RF['accuracy score']
ax = plt.gca()
max_index = np.where(accuracy_score == max(accuracy_score))
accuracy_score_max = accuracy_score[max_index[0][0]]
NumberofSelectedFeatures_max = NumberofSelectedFeatures[max_index[0][0]]
maxValue = 'features = '+str(NumberofSelectedFeatures_max)+' , '+'Accuracy = '+str(accuracy_score_max)
plt.annotate(maxValue, xy=(NumberofSelectedFeatures_max,accuracy_score_max))
ax = plt.gca()
plt.title('Feature Selection (Chi-square) of RF Model')
Chi2RF.plot(kind='line',x='Number of Selected Features',y='accuracy score',ax=ax)
plt.show()
#ปรับพารามิเตอร์
model =  RandomForestClassifier(random_state=42)

Xnew =  SelectKBest(chi2, k=5).fit_transform(X, y)
n_estimators = list(range(2, 50))
max_features = ['auto', 'sqrt']
max_depth = list(range(2, 50))
```

```
min_samples_split = list(range(2, 20))
min_samples_leaf = list(range(2, 20))
random_grid = dict(n_estimators= n_estimators,
                   max_features = max_features,
                   max_depth= max_depth,
                   min_samples_split= min_samples_split,
                   min_samples_leaf= min_samples_leaf)
rf = RandomForestClassifier(random_state=42)
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,
n_iter = 50, cv = 5,
                               verbose=2, random_state=42, n_jobs = -1)
rf_random.fit(Xnew, Y)
rf_random.best_params_
#วัดประสิทธิภาพของแบบจำลอง
clf_RFC =  RandomForestClassifier(max_depth=25,max_features='sqrt',min_samples_leaf=3,
                                  min_samples_split=9,n_estimators=38,random_state=42)
accscore = cross_val_score(clf_RFC,Xnew,target, cv=cv)
recallscore = cross_val_score(clf_RFC,Xnew,target, cv=cv,scoring='recall_macro')
pscore = cross_val_score(clf_RFC,Xnew,target, cv=cv,scoring='precision_macro')
F1_RFC = np.mean(f1score)
acc_RFC = np.mean(accscore)
recall_RFC = np.mean(recallscore)
p_RFC = np.mean(pscore)
print ('Accuracy:', acc_RFC)
print ('F1 score:', F1_RFC)
print ('Recall:', recall_RFC)
print ('Precision:', p_RFC)

selector = SelectKBest(chi2, k=5).fit(X, y)
X.columns[selector.get_support(indices=True)]

vector_names = list(X.columns[selector.get_support(indices=True)])
print(vector_names)

# Principal Component Analysis (or PCA)

#หา n_components ที่ให้ accuracy สูงสุดเมื่อสร้างแบบจำลองด้วย KNN
y=target = df.loc[:, 'diagnosis']
X=data = df[['age', 'P1', 'E2', 'F3', 'P4', 'F5', 'F6', 'F7', 'P8',
       'E9', 'E10', 'P11', 'F12', 'P13', 'F14', 'E15', 'F16', 'P17', 'E18',
       'F19', 'E20', 'E21', 'E22', 'E23', 'F24', 'P25', 'F7_E23']]
n_component = list()
accKNN = list()
for n in range(1,27):
    Xnew =  PCA(n_components=n,random_state = 42).fit_transform(X, y)
    knn_clf = KNeighborsClassifier()
    accscore = cross_val_score(knn_clf,Xnew,y, cv=5)
    acc_knn = np.mean(accscore)
    accKNN.append(acc_knn)
    n_component.append(n)

PCAKNN=pd.DataFrame({"Number of Components to Keep":n_component})
PCAKNN['accuracy score']= accKNN
PCAKNN.sort_values(by='Number of Components to Keep', ascending=1)
n_components = PCAKNN['Number of Components to Keep']
accuracy_score = PCAKNN['accuracy score']
ax = plt.gca()
max_index = np.where(accuracy_score == max(accuracy_score))
accuracy_score_max = accuracy_score[max_index[0][0]]
n_components_max = n_components[max_index[0][0]]
maxValue = 'n_components = '+str(n_components_max)+' , '+'Accuracy =
'+str(accuracy_score_max)
plt.annotate(maxValue, xy=(n_components_max-2,accuracy_score_max))
ax = plt.gca()
PCAKNN.plot(kind='line',x='Number of Components to Keep',y='accuracy score',ax=ax)
plt.show()
#ปรับพารามิเตอร์
Xnew =  PCA(n_components=8,random_state = 42).fit_transform(X, y)
knn = KNeighborsClassifier()
k_range = list(range(1, 31))
p = [1,2]
weights=['distance','uniform']
```

```
param_grid = dict(n_neighbors=k_range,p=p,weights=weights)
grid = GridSearchCV(knn, param_grid, cv=5, scoring='accuracy')
grid.fit(Xnew,Y)
print(grid.best_params_)
print(grid.best_estimator_)
#วัดประสิทธิภาพของแบบจำลอง
clf_KNN = KNeighborsClassifier(n_neighbors=22,p=1,weights='distance')
flscore = cross_val_score(clf_KNN,Xnew,target, cv=cv,scoring='f1_macro')
accscore = cross_val_score(clf_KNN,Xnew,target, cv=cv)
recallscore = cross_val_score(clf_KNN,Xnew,target, cv=cv,scoring='recall_macro')
pscore = cross_val_score(clf_KNN,Xnew,target, cv=cv,scoring='precision_macro')
f_knn = np.mean(flscore)
acc_knn = np.mean(accscore)
r_knn = np.mean(recallscore)
p_knn = np.mean(pscore)
print ('Accuracy:', acc_knn)
print ('F1 score:', f_knn)
print ('Recall:', r_knn)
print ('Precision:', p_knn)

pca = PCA(n_components=8)
fit = pca.fit(X)
pcascore = fit.explained_variance_ratio_
print("Explained Variance: ",pcascore)
print(fit.components_)

#หา n_components ที่ให้ accuracy สูงสุดเมื่อสร้างแบบจำลองด้วย GNB
accGNB = list()
for n in range(1,27):
    Xnew =  PCA(n_components=n,random_state = 42).fit_transform(X, y)
    gnb_clf = GaussianNB()
    accscore = cross_val_score(gnb_clf,Xnew,y, cv=5)
    acc_gnb = np.mean(accscore)
    accGNB.append(acc_gnb)

PCAGNB=pd.DataFrame({"Number of Components to Keep":n_component})
PCAGNB['accuracy score']= accGNB
PCAGNB.sort_values(by='Number of Components to Keep', ascending=1)
n_components = PCAGNB['Number of Components to Keep']
accuracy_score = PCAGNB['accuracy score']
ax = plt.gca()
max_index = np.where(accuracy_score == max(accuracy_score))
accuracy_score_max = accuracy_score[max_index[0][0]]
n_components_max = n_components[max_index[0][0]]
maxValue = 'n_components = '+str(n_components_max)+' , '+'Accuracy =
'+str(accuracy_score_max)
plt.annotate(maxValue, xy=(n_components_max-6,accuracy_score_max-0.005))
PCAGNB.plot(kind='line',x='Number of Components to Keep',y='accuracy score',ax=ax)
plt.show()
#วัดประสิทธิภาพของแบบจำลอง
Xnew =  PCA(n_components=17,random_state = 42).fit_transform(X, y)
gnb_clf = GaussianNB()
cv = 5
flscore = cross_val_score(gnb_clf,Xnew,target, cv=cv,scoring='f1_macro')
accscore = cross_val_score(gnb_clf,Xnew,target, cv=cv)
recallscore = cross_val_score(gnb_clf,Xnew,target, cv=cv,scoring='recall_macro')
pscore = cross_val_score(gnb_clf,Xnew,target, cv=cv,scoring='precision_macro')
f_gnb = np.mean(flscore)
acc_gnb = np.mean(accscore)
r_gnb = np.mean(recallscore)
p_gnb = np.mean(pscore)
print ('Accuracy:', acc_gnb)
print ('F1 score:', f_gnb)
print ('Recall:', r_gnb)
print ('Precision:', p_gnb)

#หา n_components ที่ให้ accuracy สูงสุดเมื่อสร้างแบบจำลองด้วย SVM
accSVC = list()
for n in range(1,27):
    Xnew =  PCA(n_components=n,random_state = 42).fit_transform(X, y)
    svc_clf = SVC(random_state=42)
    accscore = cross_val_score(svc_clf,Xnew,y, cv=5)
```

```
694         acc_svc = np.mean(accscore)
695         accSVC.append(acc_svc)
696 
697     PCASVC=pd.DataFrame({"Number of Components to Keep":n_component})
698     PCASVC['accuracy score']= accSVC
699     PCASVC.sort_values(by='Number of Components to Keep', ascending=1)
700     n_components = PCASVC['Number of Components to Keep']
701     accuracy_score = PCASVC['accuracy score']
702     ax = plt.gca()
703     max_index = np.where(accuracy_score == max(accuracy_score))
704     accuracy_score_max = accuracy_score[max_index[0][0]]
705     n_components_max = n_components[max_index[0][0]]
706     maxValue = 'n_components = '+str(n_components_max)+' , '+'Accuracy =
        '+str(accuracy_score_max)
707     plt.annotate(maxValue, xy=(n_components_max-6,accuracy_score_max-0.005))
708     PCASVC.plot(kind='line',x='Number of Components to Keep',y='accuracy score',ax=ax)
709     plt.show()
710     #ปรับพารามิเตอร์
711     Xnew =  PCA(n_components=13,random_state = 42).fit_transform(X, y)
712     C=range(1,10)
713     degree = range(1,3)
714     gamma = (0.1,5)
715     param_grid=dict(C=C,degree=degree,gamma=gamma)
716     grid=GridSearchCV(SVC(random_state=42),param_grid,cv=5)
717     grid.fit(Xnew, Y)
718     grid.best_params_
719 
720     #วัดประสิทธิภาพของแบบจำลอง
721     Xnew =  PCA(n_components=13,random_state=42).fit_transform(X, y)
722     clf_SVC = SVC( C=1,degree=1,gamma=0.1,kernel='poly',decision_function_shape='ovr',
        random_state=42)
723     cv=5
724     f1score = cross_val_score(clf_SVC,Xnew,y, cv=cv,scoring='f1_macro')
725     accscore = cross_val_score(clf_SVC,Xnew,y, cv=cv)
726     recallscore = cross_val_score(clf_SVC,Xnew,y, cv=cv,scoring='recall_macro')
727     pscore = cross_val_score(clf_SVC,Xnew,y, cv=cv,scoring='precision_macro')
728     f_svc = np.mean(f1score)
729     acc_svc = np.mean(accscore)
730     r_svc = np.mean(recallscore)
731     p_svc = np.mean(pscore)
732     print ('Accuracy:', acc_svc)
733     print ('F1 score:', f_svc)
734     print ('Recall:', r_svc)
735     print ('Precision:', p_svc)
736 
737     #หา n_components ที่ให้ accuracy สูงสุดเมื่อสร้างแบบจำลองด้วย RandomForestClassifier
738     accRF = list()
739     for n in range(1,27):
740         Xnew =  PCA(n_components=n,random_state = 42).fit_transform(X, y)
741         rf_clf =  RandomForestClassifier(random_state=42)
742         accscore = cross_val_score(rf_clf,Xnew,y, cv=5)
743         acc_rf = np.mean(accscore)
744         accRF.append(acc_rf)
745 
746     PCARF=pd.DataFrame({"Number of Components to Keep":n_component})
747     PCARF['accuracy score']= accRF
748     PCARF.sort_values(by='Number of Components to Keep', ascending=1)
749     n_components = PCARF['Number of Components to Keep']
750     accuracy_score = PCARF['accuracy score']
751     ax = plt.gca()
752     max_index = np.where(accuracy_score == max(accuracy_score))
753     accuracy_score_max = accuracy_score[max_index[0][0]]
754     n_components_max = n_components[max_index[0][0]]
755     maxValue = 'n_components = '+str(n_components_max)+' , '+'Accuracy =
        '+str(accuracy_score_max)
756     plt.annotate(maxValue, xy=(n_components_max-6,accuracy_score_max))
757     PCARF.plot(kind='line',x='Number of Components to Keep',y='accuracy score',ax=ax)
758     plt.show()
759     #ปรับพารามิเตอร์
760     Xnew =  PCA(n_components=9,random_state=42).fit_transform(X, y)
761     n_estimators = list(range(2, 50))
762     max_features = ['auto', 'sqrt']
```

```
763  max_depth = list(range(2, 50))
764  min_samples_split = list(range(2, 20))
765  min_samples_leaf = list(range(2, 20))
766  random_grid = dict(n_estimators= n_estimators,
767                     max_features = max_features,
768                     max_depth= max_depth,
769                     min_samples_split= min_samples_split,
770                     min_samples_leaf= min_samples_leaf)
771  rf = RandomForestClassifier(random_state=42)
772  rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,
     n_iter = 50, cv = 5,
773                                 verbose=2, random_state=42, n_jobs = -1)
774  rf_random.fit(Xnew, Y)
775  rf_random.best_params_
776  #วัดประสิทธิภาพของแบบจำลอง
777  Xnew =  PCA(n_components=9,random_state=42).fit_transform(X, y)
778  clf_RF = RandomForestClassifier(n_estimators=38, max_depth=31,max_features= 'auto',
779                                  min_samples_split=9,
                                    min_samples_leaf=14,random_state=42)
780  cv=5
781  f1score = cross_val_score(clf_RF,Xnew,y, cv=cv,scoring='f1_macro')
782  accscore = cross_val_score(clf_RF,Xnew,y, cv=cv)
783  recallscore = cross_val_score(clf_RF,Xnew,y, cv=cv,scoring='recall_macro')
784  pscore = cross_val_score(clf_RF,Xnew,y, cv=cv,scoring='precision_macro')
785  f_rf = np.mean(f1score)
786  acc_rf = np.mean(accscore)
787  r_rf = np.mean(recallscore)
788  p_rf = np.mean(pscore)
789  print ('Accuracy:', acc_rf)
790  print ('F1 score:', f_rf)
791  print ('Recall:', r_rf)
792  print ('Precision:', p_rf)
793
```

บรรณานุกรม

[1] K. Kroenke *et al.*, "Causes of persistent dizziness: a prospective study of 100 patients in ambulatory care," *Annals of internal medicine,* vol. 117, no. 11, pp. 898-904, 1992.

[2] S. L. Whitney, G. F. Marchetti, and L. O. Morris, "Usefulness of the dizziness handicap inventory in the screening for benign paroxysmal positional vertigo," *Otology & Neurotology,* vol. 26, no. 5, pp. 1027-1033, 2005.

[3] S. Y. Moon *et al.*, "Clinical characteristics of benign paroxysmal positional vertigo in Korea: a multicenter study," *J Korean Med Sci,* vol. 21, no. 3, pp. 539-43, Jun 2006.

[4] S.-H. Lee and J. S. Kim, "Benign paroxysmal positional vertigo," *Journal of Clinical Neurology,* vol. 6, no. 2, pp. 51-63, 2010.

[5] S. G. Korres, D. G. Balatsouras, S. Papouliakos, and E. Ferekidis, "Benign paroxysmal positional vertigo and its management," *Medical science monitor,* vol. 13, no. 6, pp. CR275-CR282, 2007.

[6] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics,* vol. 21, no. 3, pp. 660-674, 1991.

[7] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," *Statistics surveys,* vol. 4, pp. 40-79, 2010.

[8] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *bioinformatics,* vol. 23, no. 19, pp. 2507-2517, 2007.

[9] Z. M. Hira and D. F. Gillies, "A review of feature selection and feature extraction methods applied on microarray data," *Advances in bioinformatics,* vol. 2015, 2015.

[10] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature extraction: foundations and applications*. Springer, 2008.

[11] M. Von Brevern *et al.*, "Epidemiology of benign paroxysmal positional vertigo. A population-based study," *Journal of Neurology, Neurosurgery & Psychiatry,* 2006.

[12] A. Saxena and M. C. Prabhakar, "Performance of DHI score as a predictor of benign paroxysmal positional vertigo in geriatric patients with dizziness/vertigo: a cross-sectional study," *PloS one,* vol. 8, no. 3, p. e58106, 2013.

[13] W. Chen *et al.*, "Validation of 5-item and 2-item questionnaires in Chinese version of Dizziness Handicap Inventory for screening objective benign paroxysmal positional vertigo," *Neurological Sciences,* vol. 37, no. 8, pp. 1241-1246, 2016.

[14] G. P. Jacobson and C. W. Newman, *The Development of the Dizziness Handicap Inventory*. 1990, pp. 424-7.

[15] P. Harrington, *Machine learning in action*. Manning Greenwich, 2012.

ประวัติผู้เขียน

ชื่อ-สกุล น.ส.ลวณา มสารกรัณฑ์

วัน เดือน ปี เกิด 2 กันยายน 2534

ที่อยู่ปัจจุบัน 107/122 หมู่บ้านพุทธชาด ต.ท่าไม้ อ.กระทุ่มแบน จ.สมุทรสาคร 74110